

class13

Wanning Cui

```
library(BiocManager)
```

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

```
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

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

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

Attaching package: 'MatrixGenerics'

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

```
colAlls, colAnyNAs, colAnys, colAvgPerRowSet, 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, rowAvgPerColSet,
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
```

```
##Data import
```

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

Q1. How many genes are in this dataset?

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG0000000000419	467	523	616	371	582
ENSG0000000000457	347	258	364	237	318
ENSG0000000000460	96	81	73	66	118
ENSG0000000000938	0	0	1	0	2

	SRR1039517	SRR1039520	SRR1039521
ENSG000000000003	1097	806	604
ENSG000000000005	0	0	0
ENSG0000000000419	781	417	509
ENSG0000000000457	447	330	324
ENSG0000000000460	94	102	74
ENSG0000000000938	0	0	0

```
nrow(counts)
```

```
[1] 38694
```

Q2. How many 'control' cell lines do we have?

```
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

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

```
[1] 4
```

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

```
control treated  
      4      4
```

```
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
```

I want to compare the control to the treated columns. To do this, I will

-Step 1. Identify and extract the “control” columns. -Step 2. Calculate the mean value per gene for all these “control” columns and save as `control.mean` -Step 3. Do the same for treated -Step 4. Compare the `control.mean` and `treated.mean` values

Step1:

```
control.inds <- metadata$dex=="control"
```

```
metadata[control.inds,]
```

```
      id      dex celltype      geo_id  
1 SRR1039508 control   N61311 GSM1275862  
3 SRR1039512 control   N052611 GSM1275866  
5 SRR1039516 control   N080611 GSM1275870  
7 SRR1039520 control   N061011 GSM1275874
```

```
head(counts[,control.inds])
```

	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

```
control.mean <- rowMeans(counts[,control.inds])
head(control.mean)
```

ENSG000000000003	ENSG000000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
900.75	0.00	520.50	339.75	97.25
ENSG000000000938				
0.75				

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

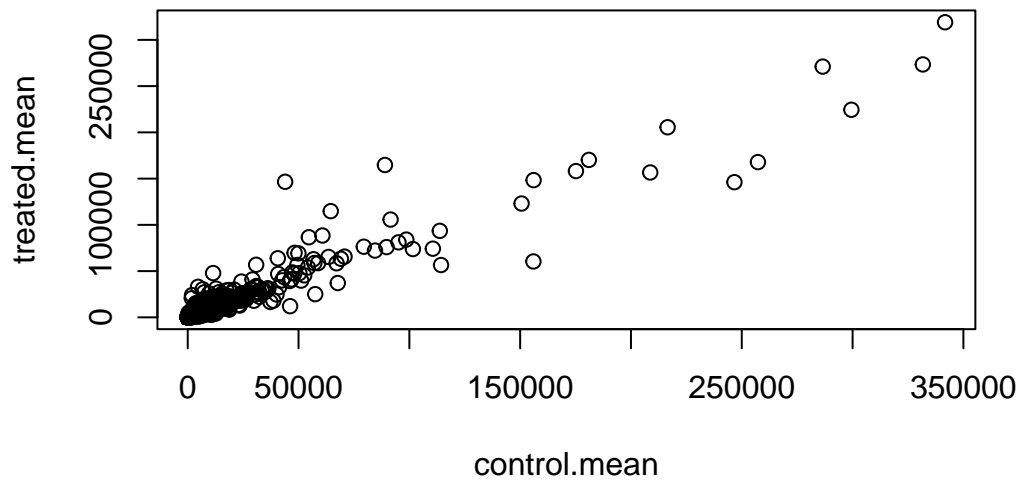
ENSG000000000003	ENSG000000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
658.00	0.00	546.00	316.50	78.75
ENSG000000000938				
0.00				

We will combine our mean data counts

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

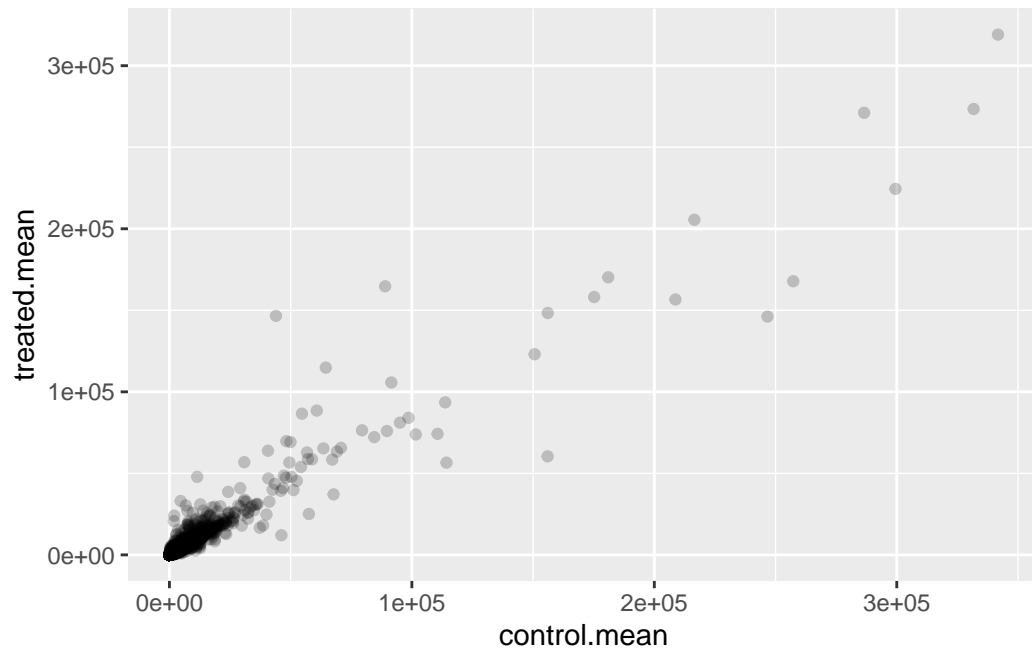
Let's see what these count values look like

```
plot(meancounts)
```



```
library(ggplot2)

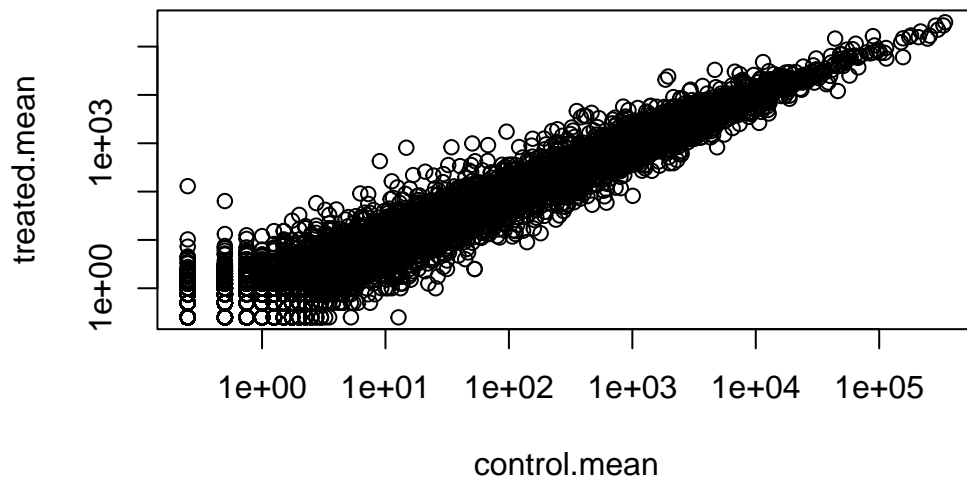
ggplot(meancounts)+
  aes(control.mean,treated.mean)+
  geom_point(alpha=0.2)
```



```
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



Logs are super useful when we have such skewed data

```
#Treated/control
log2(10/10)
```

```
[1] 0
```

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

```
[1] 1
```

Add log2(Fold-change) values to our wee results table.

```
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

I need to exclude any genes with zero counts as we can't say anything about them anyways from this experiment and it causes me math pain.

```
# What values in the first two cols are zero
to.rm.inds <- rowSums(meancounts[,1:2]==0) > 0
mycounts <- meancounts[!to.rm.inds,]
```

Q. How many genes do I have left

```
nrow(mycounts)
```

[1] 21817

Q. How many genes are “up regulated” i.e. have a $\log_2(\text{fold-change})$ greater than +2?

```
sum(mycounts$log2fc > +2)
```

[1] 250

Q. How many genes are “down regulated”?

```
sum(mycounts$log2fc < -2)
```

[1] 367

Q. We don't trust the data results because there are a lot of missing information

##Running DESeq

Like many bioconductor analysis packages DESeq wants its input in a very particular way.

```
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

To run DESeq analysis, we call the main function from the package called 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 this dds object, we can use the DESeq results() function.

```
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

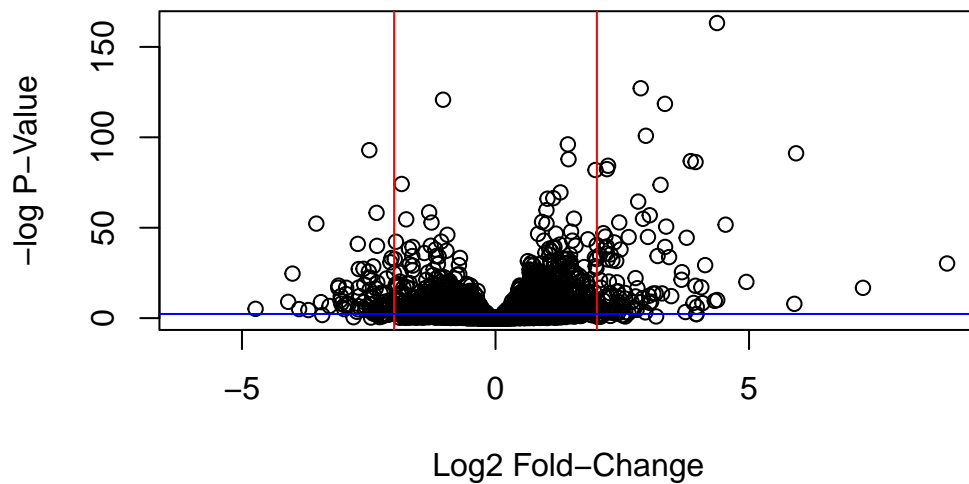
	baseMean	log2FoldChange	lfcSE	stat	pvalue
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
ENSG000000000003	747.194195	-0.3507030	0.168246	-2.084470	0.0371175
ENSG000000000005	0.000000	NA	NA	NA	NA
ENSG000000000419	520.134160	0.2061078	0.101059	2.039475	0.0414026
ENSG000000000457	322.664844	0.0245269	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				

A common summary visualization is called a Volcano plot.

```
plot(res$log2FoldChange, -log(res$padj),
     xlab="Log2 Fold-Change",
     ylab="-log P-Value")

abline(v=c(-2,2), col="red")
abline(h=-log(0.1), col="blue")
```



```
mycols <- rep("gray",nrow(res))
mycols
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

[38661] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38671] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38681] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38691] "gray" "gray" "gray" "gray"

```

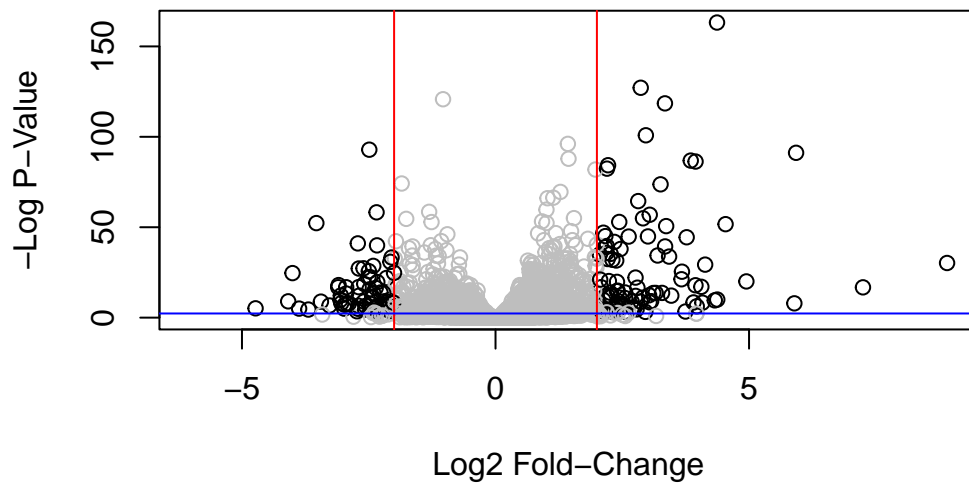
```

mycols <- rep("gray",nrow(res))
mycols[res$log2FoldChange > 2] <- "black"
mycols[res$log2FoldChange < -2] <- "black"
mycols[res$padj > 0.05] <- "gray"

plot(res$log2FoldChange, -log(res$padj), col=mycols,
      xlab="Log2 Fold-Change",
      ylab="-Log P-Value")

abline(v=c(-2,2), col="red")
abline(h=-log(0.1), col="blue")

```



Save our results to date

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

##Adding annotation data

We need to translate or “map” our ensemble IDs into more understandable gene names and the identifiers that other useful databases use.

```
library(AnnotationDbi)
```

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

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

```
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"				

```
res$symbol <- mapIds(org.Hs.eg.db,  
                     keys=row.names(res), #Our genenames  
                     keytype="ENSEMBL",  
                     column="SYMBOL",  
                     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			

```
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

```
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$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

```
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		genename
	<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

```
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"
```

```
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
```

Run gage:

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

```
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 have a look at one of these pathways

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

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

Info: Working in directory D:/UCSD/R/class13

Info: Writing image file hsa05310.pathview.png

