

class 13

Kalodiah Toma

```
library(DESeq2)
```

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 (Himes et al. 2014).

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

```
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
```

```
nrow(counts)
```

```
[1] 38694
```

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

```
[1] 4
```

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

```
control treated
4          4
```

Q1. How many genes are in this dataset? 38694 Q2. How many ‘control’ cell lines do we have?
4

I want to compare the control to the treated columns. To do this I will: 1. identify and extract the control columns from the metadata 2. calculate the mean value per gene of the control columns and save as ‘control.mean’ 3. do the same for the treated columns 4. compare the ‘control.mean’ and ‘treated.mean’ values

Step 1:

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

```
ENSG000000000003 ENSG000000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75          0.00        520.50        339.75        97.25
ENSG000000000938
         0.75
```

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

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460  
658.00 0.00 546.00 316.50 78.75  
ENSG00000000938  
0.00
```

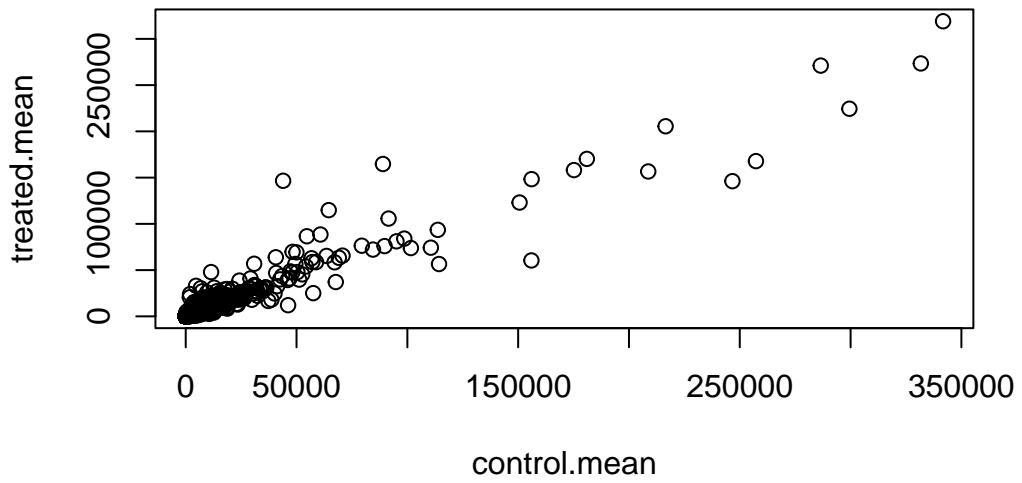
```
#if you wrote the code in one step  
treated.mean<-rowMeans(counts[,metadata$dex=="treated"])
```

Q3. How would you make the above code in either approach more robust? Is there a function that could help here? Because using “/4” will only give you values if you have four values. rowMeans will give you the means regardless of the number of values.

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) #treated.mean<-rowMeans(counts[,metadata\$dex==“treated”])

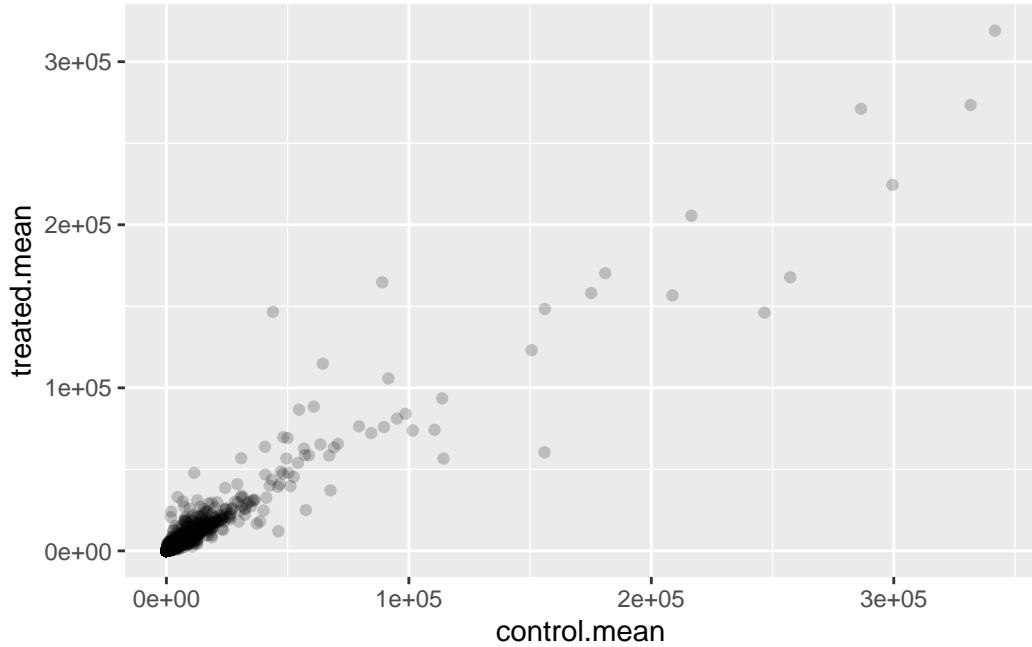
Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

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

```
library(ggplot2)
ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  geom_point(alpha=0.2)
```

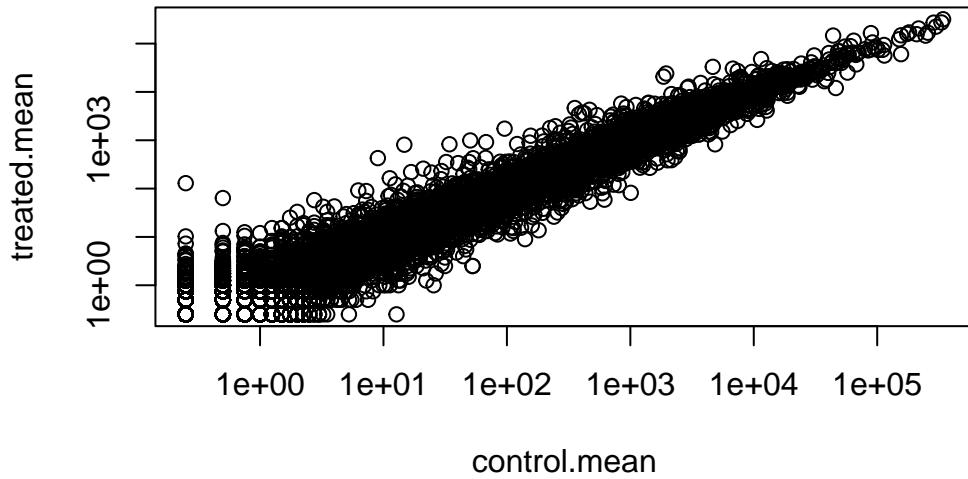


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this? #plot(meancounts, log="xy")

```
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 they are also handy when we are most interested in orders of magnitude change

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

```
[1] 0
```

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

```
[1] 1
```

```
log2(5/10)
```

```
[1] -1
```

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

```
[1] 2
```

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

```
meancounts$log2fc<-log2(meancounts$treated/meancounts$control)
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 anyway from this experiment and it causes me math pain.

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

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=TRUE argument will cause which() to return both the row and column indices (i.e. positions) where there are TRUE values. In this case this will tell us which genes (rows) and samples (columns) have zero counts. We are going to ignore any genes that have zero counts in any sample so we just focus on the row answer. Calling unique() will ensure we don't count any row twice if it has zero entries in both samples.

How many genes do T have left? 21817

```
nrow(mycounts)
```

```
[1] 21817
```

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

```
[1] 250
```

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

```
[1] 367
```

Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level? 250

Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level? 367

Q10. Do you trust these results? Why or why not? Is this data statistically significant? However, fold change can be large (e.g. »two-fold up- or down-regulation) without being statistically significant (e.g. based on p-values). We have not done anything yet to determine whether the differences we are seeing are significant.

```
##Running DESeq2
```

Like many bioconductor packages DESeq2 wants it's 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)'

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

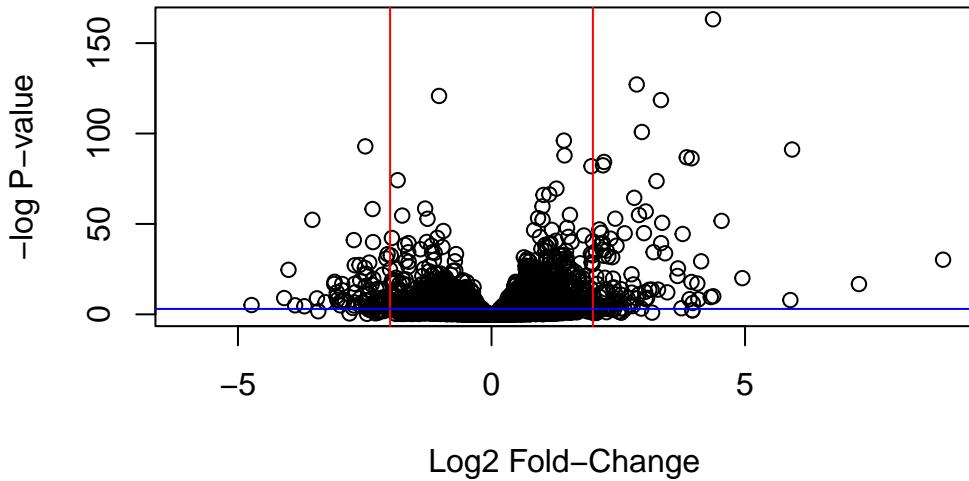
To get the results out of this ‘dds’ object we can use the DESeq ‘results()’ function.

```
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
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
```

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(0.05), col="blue")
```



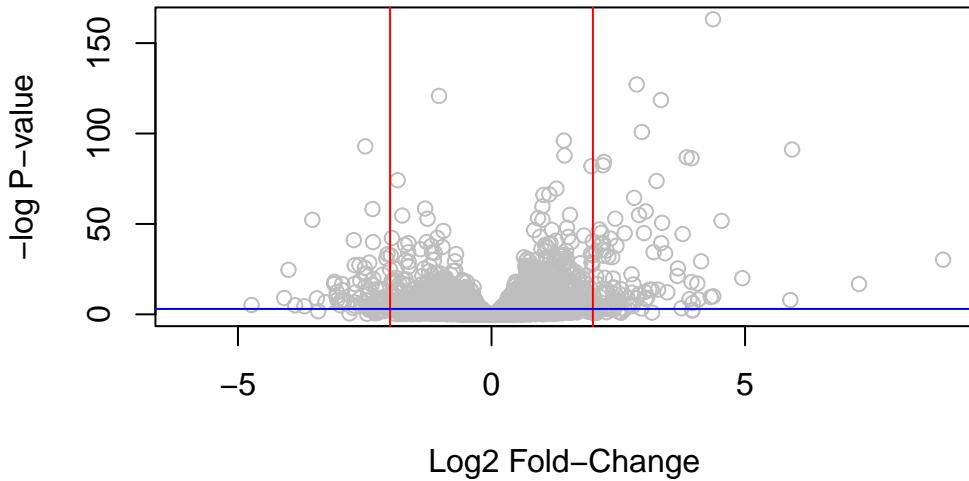
```
mycols<-rep("gray", nrow(res))
mycols
```

```
[1] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[11] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[21] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[31] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[41] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[51] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[61] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[71] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[81] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[91] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[101] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[111] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[121] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[131] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[141] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[151] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[161] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[171] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[181] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
```



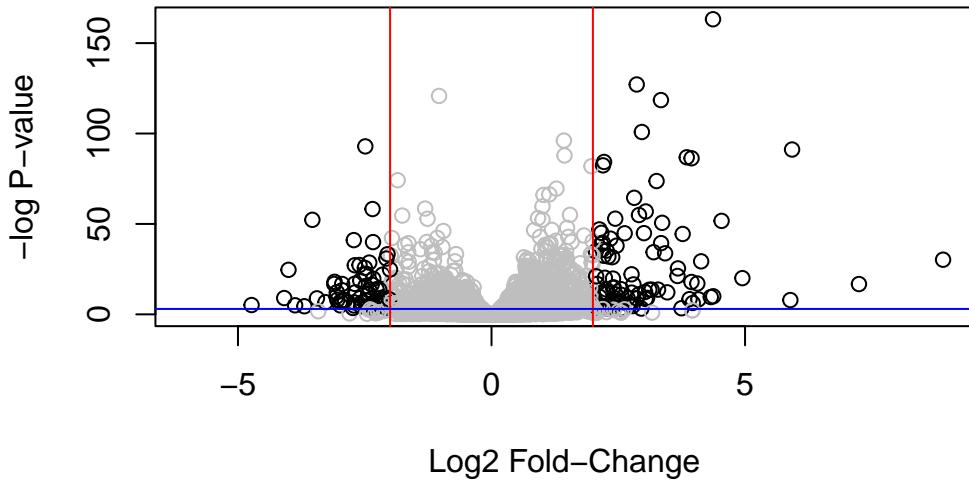
```
[38461] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38471] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38481] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38491] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38501] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38511] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38521] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38531] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38541] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38551] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38561] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38571] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38581] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38591] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38601] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38611] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38621] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38631] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38641] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[38651] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[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"
```

```
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(0.05), col="blue")
```



```
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(0.05), col="blue")
```



```
#save our results to data
```

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

```
#Adding annotation data
```

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

```
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

```

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), # 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$uniport <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",   # The format of our genenames
                      column="UNIPROT",    # 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	uniport	
	<numeric>	<character>	<character>	<character>	
ENSG000000000003	0.163035	TSPAN6	7105	AOA024RCI0	
ENSG000000000005	NA	TNMD	64102	Q9H2S6	
ENSG000000000419	0.176032	DPM1	8813	060762	
ENSG000000000457	0.961694	SCYL3	57147	Q8IZE3	
ENSG000000000460	0.815849	FIRRM	55732	AOA024R922	
ENSG000000000938	NA	FGR	2268	P09769	

```

res$genename <- 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 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	uniport	
	<numeric>	<character>	<character>	<character>	
ENSG000000000003	0.163035	TSPAN6	7105	AOA024RCI0	
ENSG000000000005	NA	TNMD	64102	Q9H2S6	
ENSG000000000419	0.176032	DPM1	8813	060762	
ENSG000000000457	0.961694	SCYL3	57147	Q8IZE3	

```
ENSG00000000460 0.815849      FIRRM      55732 AOA024R922
ENSG00000000938 NA          FGR       2268    P09769
                                genename
                                <character>
ENSG00000000003      tetraspanin 6
ENSG00000000005      tenomodulin
ENSG000000000419 dolichyl-phosphate m..
ENSG000000000457 SCY1 like pseudokina..
ENSG000000000460 FIGNL1 interacting r..
ENSG00000000938 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  

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
```

lets 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 /Users/kalodiahtoma/Desktop/phd 2023/bggm 213/class 13

Info: Writing image file hsa05310.pathview.png

```
pathview(gene.data=foldchanges, pathway.id="hsa05332")
```

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

Info: Working in directory /Users/kalodiahtoma/Desktop/phd 2023/bggm 213/class 13

Info: Writing image file hsa05332.pathview.png

```
pathview(gene.data=foldchanges, pathway.id="hsa04940")
```

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

Info: Working in directory /Users/kalodiahtoma/Desktop/phd 2023/bggm 213/class 13

Info: Writing image file hsa04940.pathview.png

