

Class 13 RNA Seq Analysis with DESeq2

Irene Hsieh (A16197563)

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 "count data" and "col data".

Data Explore

```
# Complete the missing code
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
ENSG000000000419	467	523	616	371	582
ENSG000000000457	347	258	364	237	318
ENSG000000000460	96	81	73	66	118
ENSG000000000938	0	0	1	0	2
	SRR1039517	SRR1039520	SRR1039521		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG000000000419	781	417	509		
ENSG000000000457	447	330	324		
ENSG000000000460	94	102	74		
ENSG000000000938	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
```

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

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

```
[1] 4
```

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 count data.

```
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`.

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

```
#Where does it tell me which columns are control?
```

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

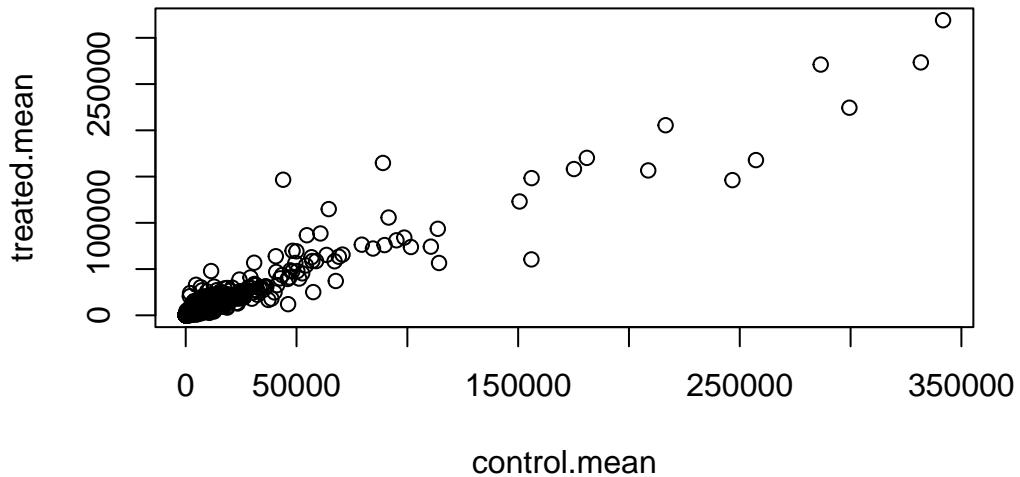
Now do the same for the treated samples to get `treated.mean` > Q4. Follow the same procedure for the treated samples (i.e. calculate the mean per gene across drug treated samples and assign to a labeled vector called `treated.mean`)

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

Have a 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?

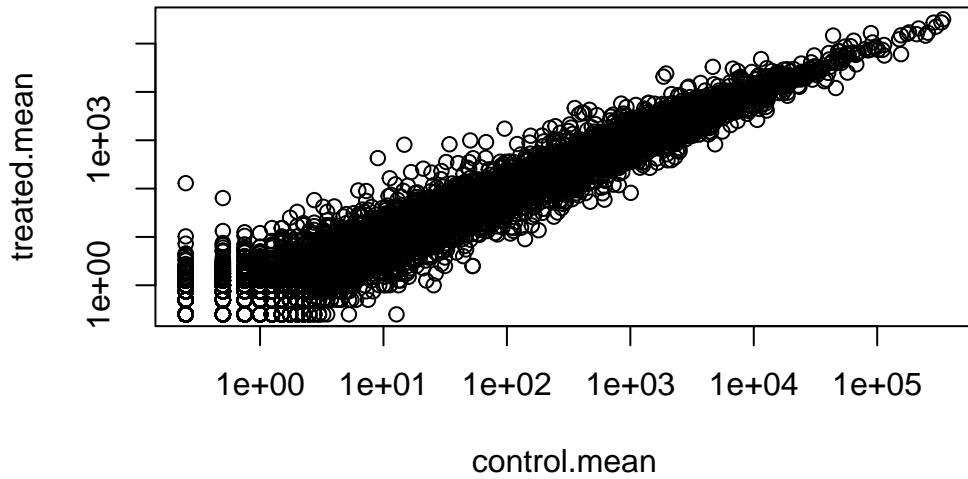
```
geom_point()
```

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
```



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
```

```
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

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

```
[1] 1846
```

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

A doubling in the treated:

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

```
[1] 1
```

```
log2(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.

we first need to remove zero count genes - as we cant say anything about these genes anyway and their division of log values are messing things up (divided by zero) or the -infinity log problem

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
```

```
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

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?

arr.ind argument in the which() function identifies the indices where either the first or second column of meancounts is equal to zero. We take the first column of the column to ensure that if there are duplicates occurring within the codes

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

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

```
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

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
```

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

no, because the results may not be significant enough to validate the results. We are missing stats for 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, colAnyNs, 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, rowAnyNs, rowAvgsPerColSet,
  rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
  rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
  rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
```

```
rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,  
rowSdDiff, rowSds, rowSums2, rowTabulates, rowVarDiff, 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
```

The 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)

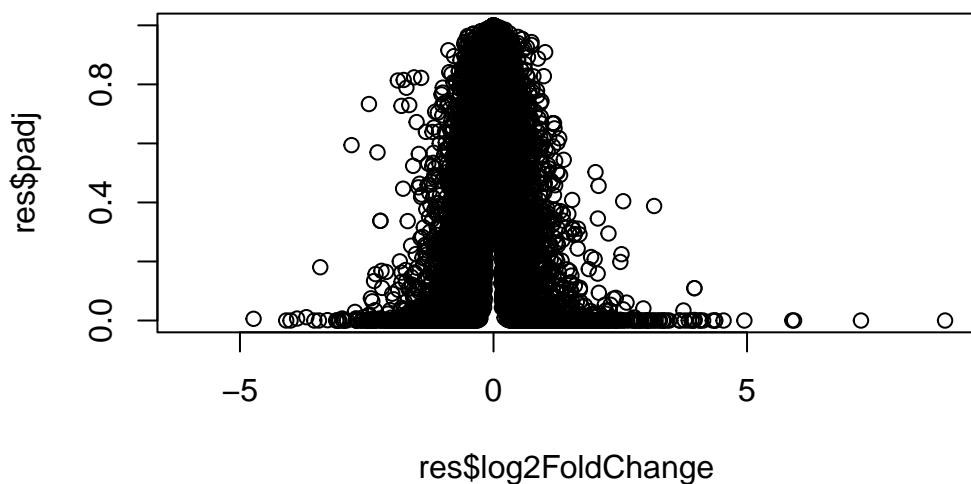
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
  baseMean log2FoldChange      lfcSE      stat     pvalue
  <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.3507030 0.168246 -2.084470 0.0371175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

```

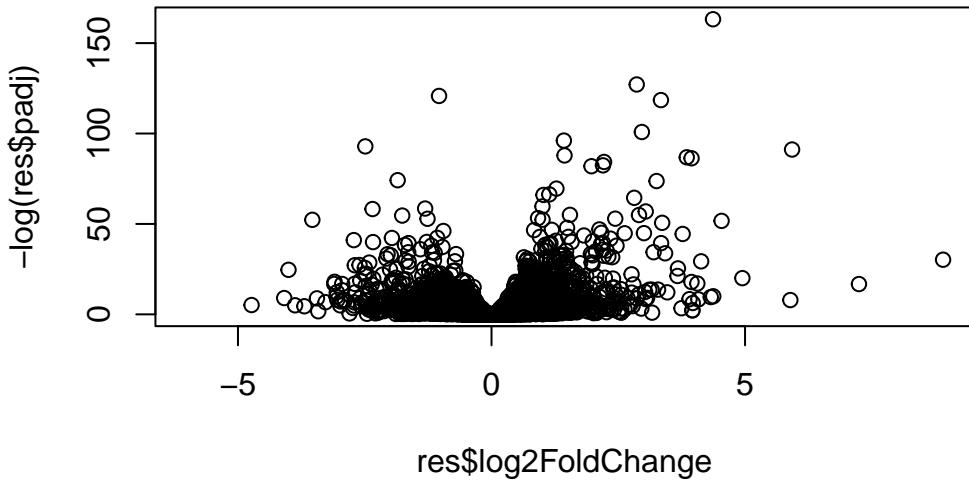
Volcano plot

A very common and useful summary results from 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))
```



The smaller the log, the more significant the magnitude, the negative in front of the log flip the volcano back from upside down

```
log(0.00005)
```

```
[1] -9.903488
```

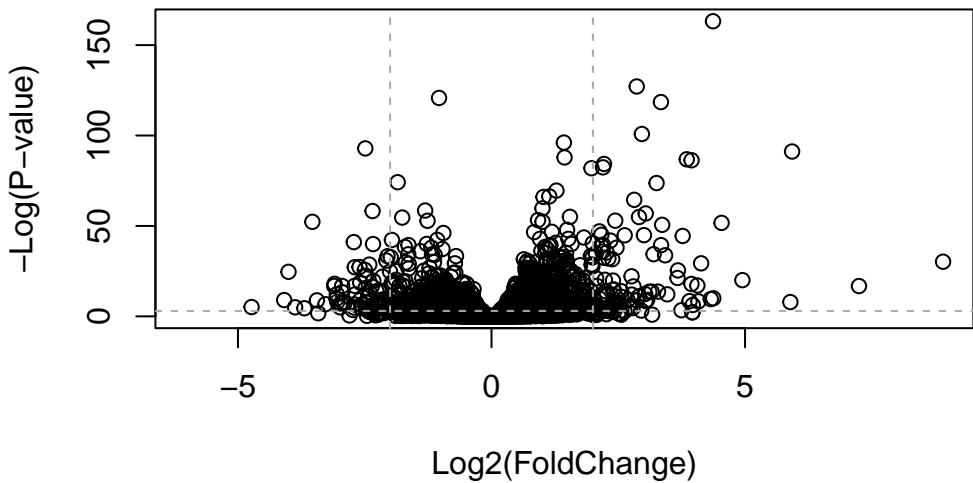
```
log(0.5)
```

```
[1] -0.6931472
```

Add some color and nice

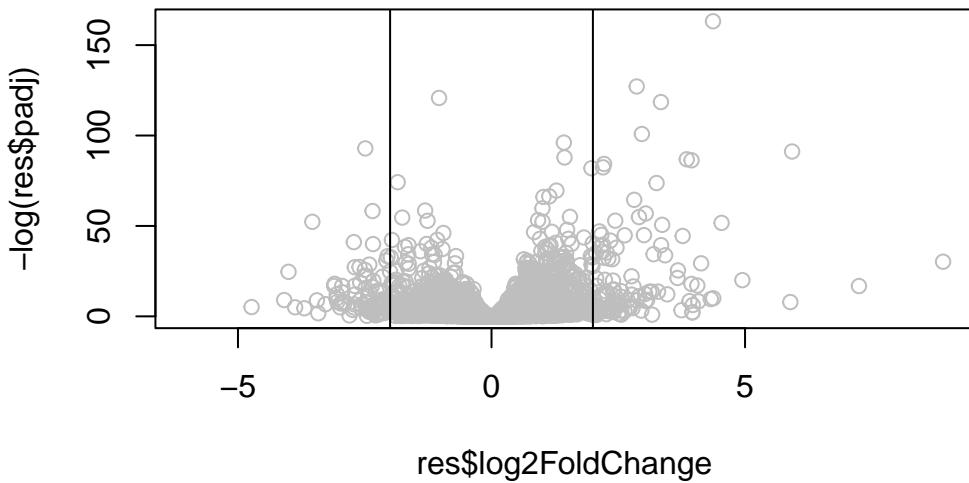
```
plot( res$log2FoldChange, -log(res$padj),
      ylab="-Log(P-value)", xlab="Log2(FoldChange)")

# Add some cut-off lines
abline(v=c(-2,2), col="darkgray", lty=2)
abline(h=-log(0.05), col="darkgray", lty=2)
```

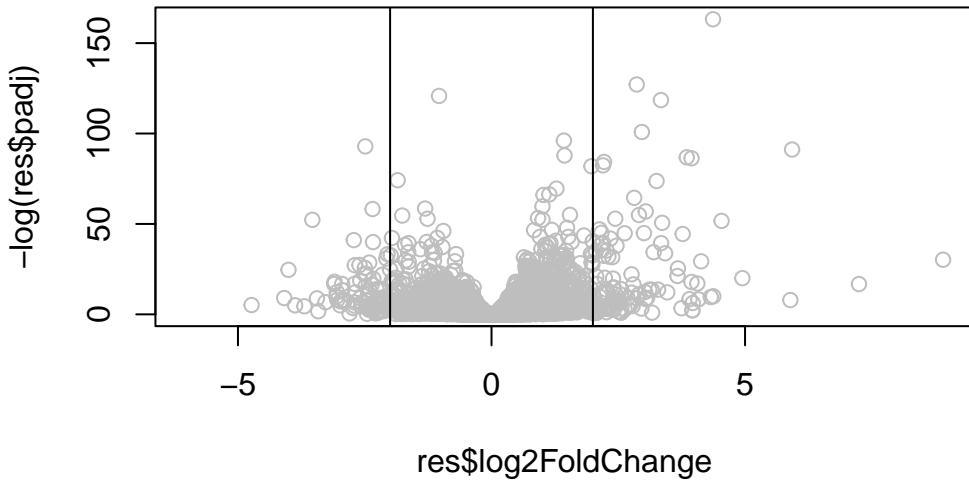


```
mycols <- rep("gray", nrow(res))

plot( res$log2FoldChange, -log(res$padj), col = mycols)
abline(v=c(-2,+2))
```



```
mycols [res$log2FoldChange >2 ] <- "blue"  
mycols [res$log2FoldChange >0.02 ] <- "grey"  
  
plot(res$log2FoldChange, -log(res$padj), col=mycols)  
abline(v=c(-2,+2))
```



```

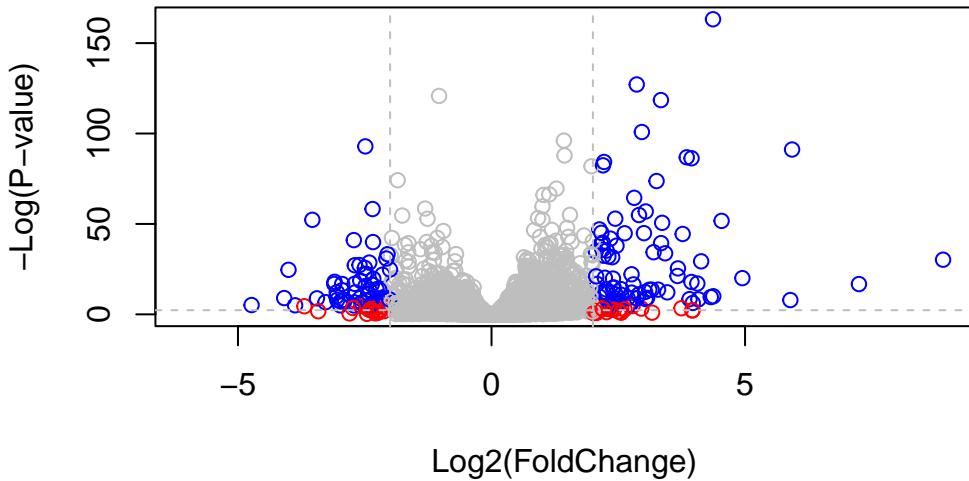
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

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)" )

# Cut-off lines
abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)

```



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
```

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"

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

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

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

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

res$genename <- mapIds(org.Hs.eg.db,
                      keys=row.names(res),
                      column="GENENAME",
                      keytype="ENSEMBL",
                      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
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      symbol      entrez      uniprot
  <numeric> <character> <character> <character>
ENSG000000000003 0.163035    TSPAN6      7105 AOA024RCI0
ENSG000000000005 NA          TNMD       64102 Q9H2S6
ENSG00000000419 0.176032    DPM1       8813  D60762
ENSG00000000457 0.961694    SCYL3      57147 Q8IZE3
ENSG00000000460 0.815849    FIRRM      55732 AOA024R922
ENSG00000000938 NA          FGR        2268  P09769
  genename
  <character>
ENSG000000000003      tetraspanin 6
ENSG000000000005      tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 FIGNL1 interacting r..
ENSG00000000938 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(a.k.a. pathway analysis, geneset enrichment, overlap analysis)

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)
```

```
# 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 the analysis

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

I can now use the returned pathway 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/chia-chianhsieh/Desktop/BIMM 143 R/Class 13

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

