

Transcriptomics and the analysis of RNA-Seq data

Kiley Hooker (PID: A15441609)

2/22/2022

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).

Read the countdata and coldata

We need two things the countdata and coldata (metadata about how and experiment was setup).

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

Have a little look see:

```
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? 38694

Q2. How many ‘control’ cell lines do we have? 4

I always need to double check that the columns of my countdown coldata (metadata) match.

```
metadata$id  
  
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"  
## [6] "SRR1039517" "SRR1039520" "SRR1039521"  
  
colnames(counts)  
  
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"  
## [6] "SRR1039517" "SRR1039520" "SRR1039521"  
  
metadata$id == colnames(counts)  
  
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
  
all(c(T, T, T, F))  
  
## [1] FALSE
```

I can use the `all()` function to make sure all my values match (i.e. all values are TRUE)

```
all(metadata$id == colnames(counts))  
  
## [1] TRUE
```

2 Extract control and treated counts for comparison

First lets extract the control counts columns

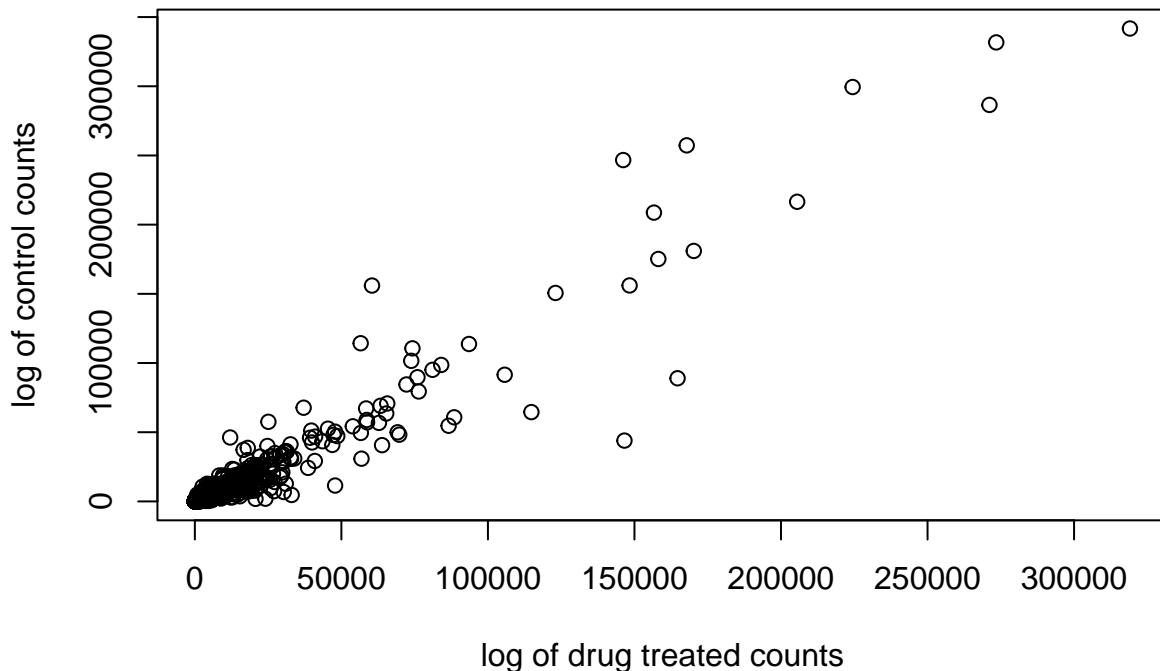
```
control.ids <- metadata[metadata$dex == "control", ]$id  
control.counts <- counts[,control.ids]  
#head(control.counts)  
  
# Take the mean count value per gene (i.e. row)  
control.mean <- rowMeans(control.counts)  
head(control.mean)  
  
## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460  
## 900.75 0.00 520.50 339.75 97.25  
## ENSG00000000938  
## 0.75
```

Now do the same thing for “treated” samples.

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

Now we can make a plot comparing treated vs control

```
plot(treated.mean, control.mean,
      xlab= "log of drug treated counts",
      ylab= "log of control counts")
```



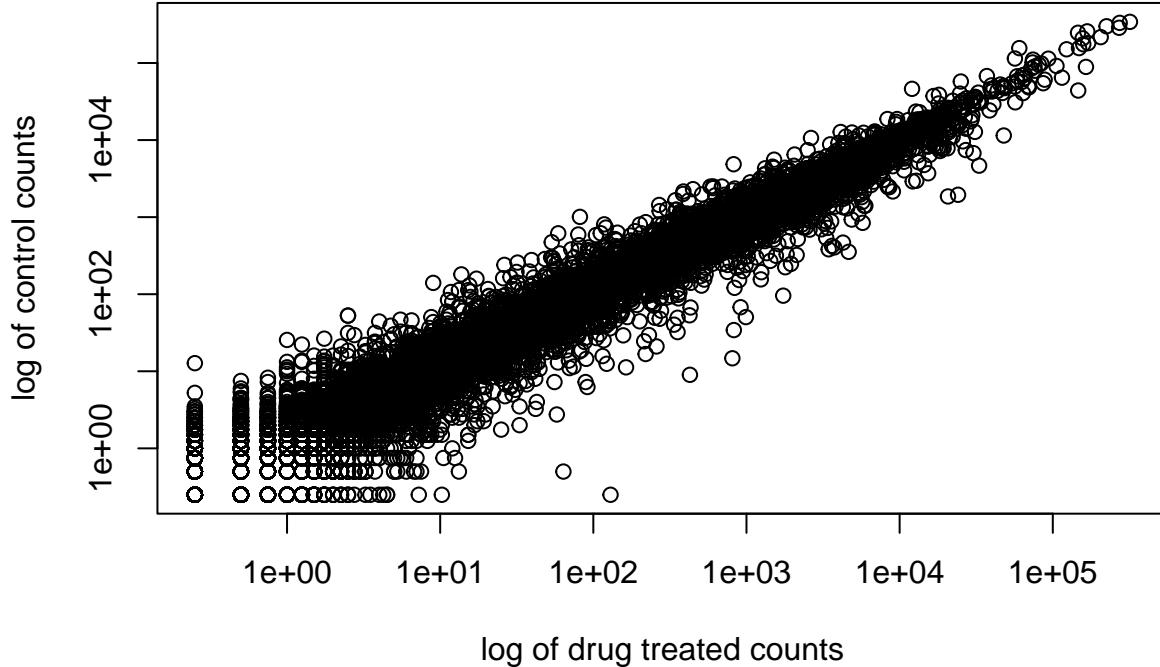
Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot? geom_point

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

```
plot(treated.mean, control.mean, log = "xy",
      xlab= "log of drug treated counts",
      ylab= "log of control counts")
```

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

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 y values <= 0 omitted  
## from logarithmic plot
```



We are after changes in gene expression: treated vs control and this would represent points (i.e. genes) that do not lie on the diagonal.

We like to work with log 2 values

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

```
## [1] 0
```

```
# doubling  
log2(40/20)
```

```
## [1] 1
```

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

```
## [1] -1
```

```
# quadrupling  
log2(80/20)
```

```
## [1] 2
```

Now lets calculate the log2 fold change

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

Store my work so far

```
meancounts <- data.frame(control.mean, treated.mean, log2fc)
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=TRUE)
```

```
## [1] 1846
```

Filter our data to remove genes with zero expression values.

```
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
```

How many genes do we have left?

```
nrow(mycounts)
```

```
## [1] 21817
```

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? It shows the row and column position where values of 0 are true.

A common threshold used for calling something differentially expressed is a log2(FoldChange) of greater than 2 or less than -2. Let's filter the dataset both ways to see how many genes are up or down-regulated.

```
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-2)
table(up.ind)
```

```
## up.ind
## FALSE TRUE
## 21567 250
```

```
table(down.ind)
```

```
## down.ind
## FALSE TRUE
## 21450 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? No, not enough analysis has been done to determine if the results are significant.

We are missing the stats! Are these differences significant?

DESeq2 analysis

Let's do this the right way. DESeq2 is an R package specifically for analyzing count-based NGS data like RNA-seq. It is available from Bioconductor.

```
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, 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 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

```

This package wants input in a specific 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

```

```
dds
```

```

## class: DESeqDataSet
## dim: 38694 8
## metadata(1): version
## assays(1): counts
## rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
##   ENSG00000283123
## rowData names(0):
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(4): id dex celltype geo_id

```

Now we can run the DESeq2 analysis

```

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 back in a useful way we can use the `results()` function

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

```

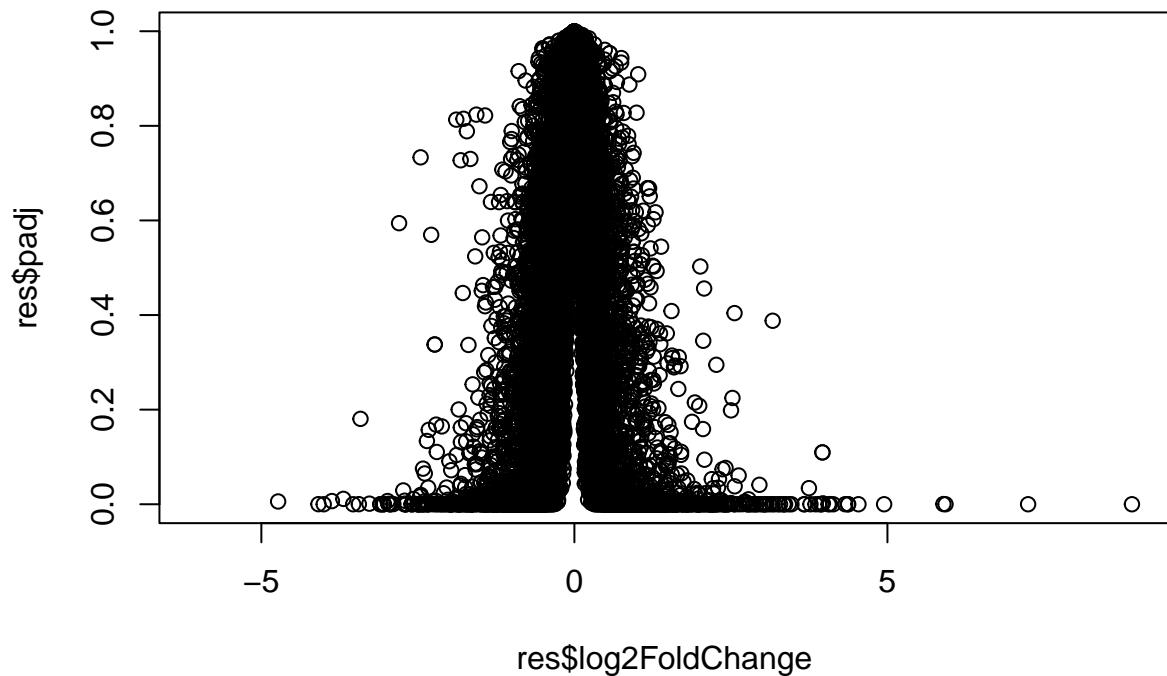
## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 38694 rows and 6 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
## <numeric>      <numeric>      <numeric>      <numeric>      <numeric>
## ENSG000000000003  747.1942    -0.3507030   0.168246  -2.084470  0.0371175
## ENSG000000000005   0.0000       NA         NA         NA         NA
## ENSG000000000419  520.1342    0.2061078   0.101059   2.039475  0.0414026
## ENSG000000000457  322.6648    0.0245269   0.145145   0.168982  0.8658106
## ENSG000000000460  87.6826    -0.1471420   0.257007  -0.572521  0.5669691
## ...
## ...
## ENSG00000283115  0.000000       NA         NA         NA         NA
## ENSG00000283116  0.000000       NA         NA         NA         NA
## ENSG00000283119  0.000000       NA         NA         NA         NA
## ENSG00000283120  0.974916    -0.668258   1.69456  -0.394354  0.693319
## ENSG00000283123  0.000000       NA         NA         NA         NA
##           padj
## <numeric>
## ENSG000000000003  0.163035
## ENSG000000000005   NA
## ENSG000000000419  0.176032
## ENSG000000000457  0.961694
## ENSG000000000460  0.815849
## ...
## ...
## ENSG00000283115   NA
## ENSG00000283116   NA
## ENSG00000283119   NA
## ENSG00000283120   NA
## ENSG00000283123   NA

```

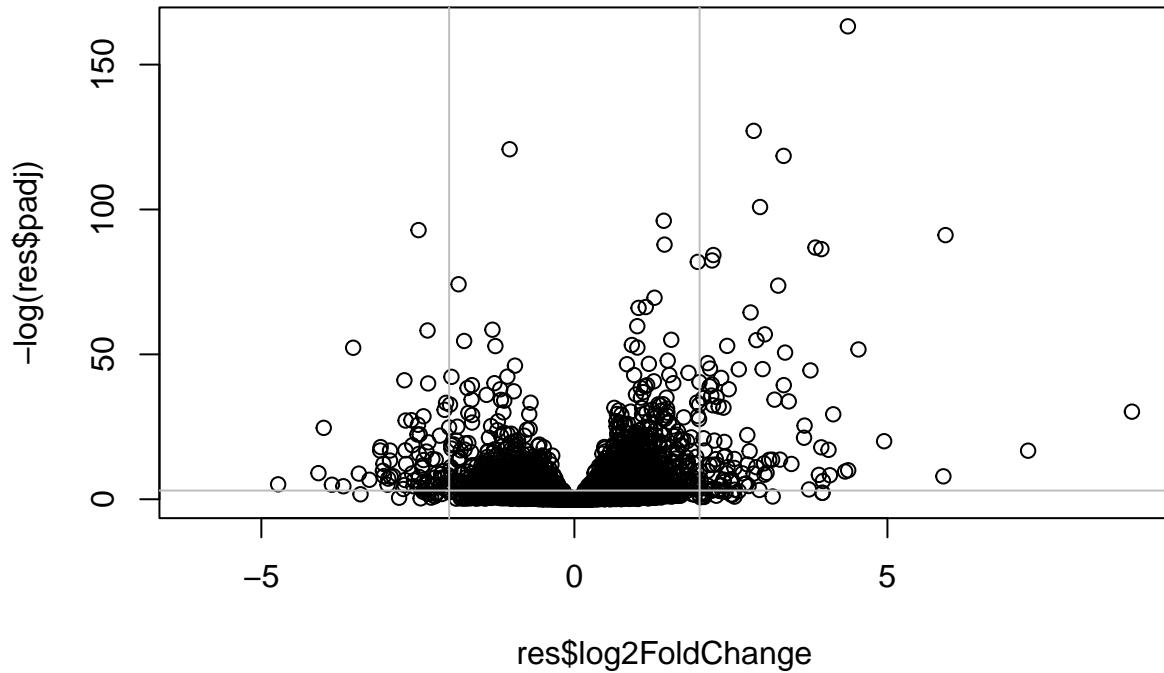
Volcano plots

Let's make a commonly produced visualization from this data, namely a so-called Volcano plot. These summary figures are frequently used to highlight the proportion of genes that are both significantly regulated and display a high fold change.

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



```
plot(res$log2FoldChange, -log(res$padj))
abline(h=-log(0.05), col="gray")
abline(v=c(-2,2), col="gray")
```



I want to polish this main results figure by adding color to the genes I will focus on next day.

```

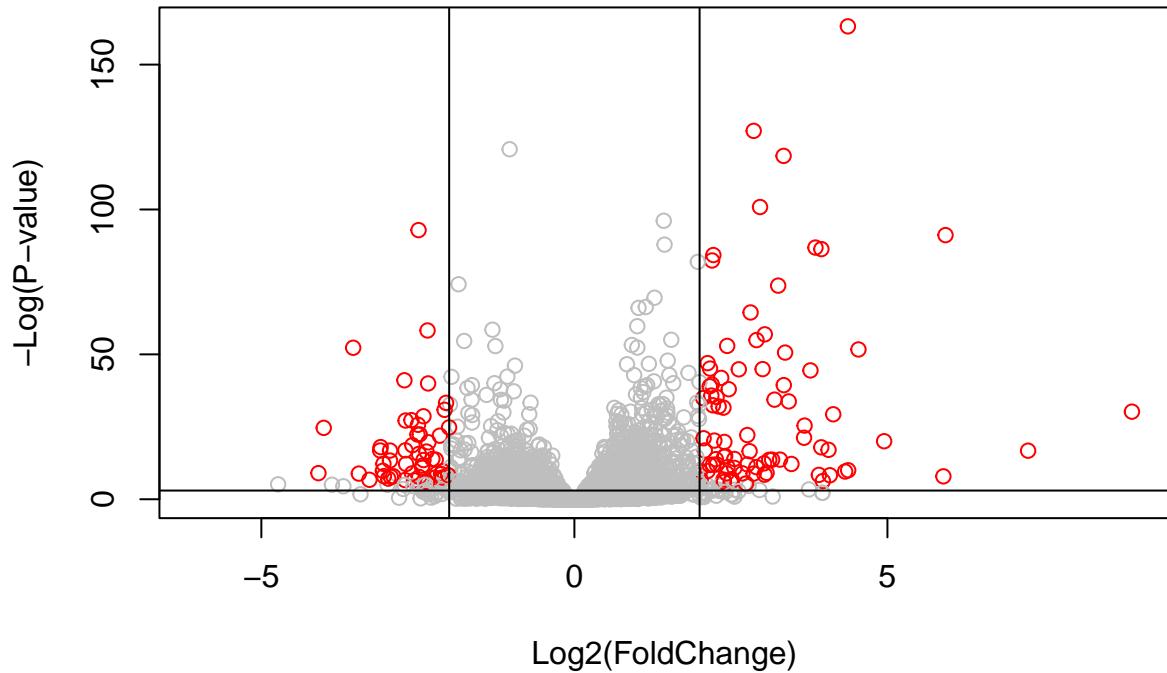
# I will start by making a gray vector for everything
mycols <- rep("gray", nrow(res))

# Now I will overwrite the small padj values
mycols[res$padj < 0.005] <- "red"

# Now if my log2foldchange is small I will make them gray
mycols[ abs(res$log2FoldChange) < 2 ] <- "gray"

plot(res$log2FoldChange, -log(res$padj), col=mycols,
     ylab="-Log(P-value)", xlab="Log2(FoldChange)")
abline(h=-log(0.05))
abline(v=c(-2,2))

```



Adding annotation data

To help interpret our results we need to understand what the differentially expressed genes are. A first step here is to get the gene names (i.e. gene SYMBOLs).

For this I will install:

- BiocManager::install("AnnotationDbi")
- BiocManager::install("org.Hs.eg.db")

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

```
##
```

What DB identifiers can I look up?

```
columns(org.Hs.eg.db)
```

```
## [1] "ACCPNUM"      "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"
```

We will use the `mapIds()` function to translate between different ids.

```
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
## 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
##           <numeric> <character>
## ENSG000000000003 0.163035    TSPAN6
## ENSG000000000005  NA        TNMD
## ENSG00000000419  0.176032    DPM1
## ENSG00000000457  0.961694    SCYL3
## ENSG00000000460  0.815849    C1orf112
## ENSG00000000938  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),
                      keytype="ENSEMBL",
                      column="ENTREZID",
                      multiVals="first")

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

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

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

```

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 10 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
##           <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG00000000003 747.194195 -0.3507030  0.168246 -2.084470 0.0371175
## ENSG00000000005  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>
## ENSG00000000003 0.163035    TSPAN6      7105 AOA024RCIO
## ENSG00000000005   NA        TNMD      64102 Q9H2S6
## ENSG00000000419 0.176032    DPM1      8813 060762
## ENSG00000000457 0.961694    SCYL3      57147 Q8IZE3
## ENSG00000000460 0.815849    C1orf112  55732 AOA024R922
## ENSG00000000938   NA        FGR       2268 P09769
##           genename
##           <character>
## ENSG00000000003      tetraspanin 6
## ENSG00000000005      tenomodulin
## ENSG00000000419 dolichyl-phosphate m..
## ENSG00000000457 SCY1 like pseudokina..
## ENSG00000000460 chromosome 1 open re..
## ENSG00000000938 FGR proto-oncogene, ..

```

Pathway Analysis with R and Bioconductor

Here we use the GAGE package (which stands for Generally Applicable Gene set Enrichment), to do KEGG pathway enrichment analysis on our RNA-seq based differential expression results.

I need to install the gage package along with the pathview package for generating pathway figures from my results.

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

Now load up the packages and have a peak at the first two pathways in KEGG

```
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"


```

Recall that vectors can have a names attribute that helps with bookkeeping just like colnames and rownames.

```

x <- c(40,70,20)
names(x) <- c("lisa", "xinqiu", "barry")
x

##    lisa xinqiu barry
##      40      70      20

```

We need a vector of fold-change labeled with the names of our genes in ENTREZ format.

```

foldchanges = res$log2FoldChange
names(foldchanges) = res$entrez

```

Now we can run the GAGE analysis passing in our foldchange vector and the KEGG genesets we are interested in.

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

Let's have a look at what is contained in this keggres results object (i.e. it's attributes).

```
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

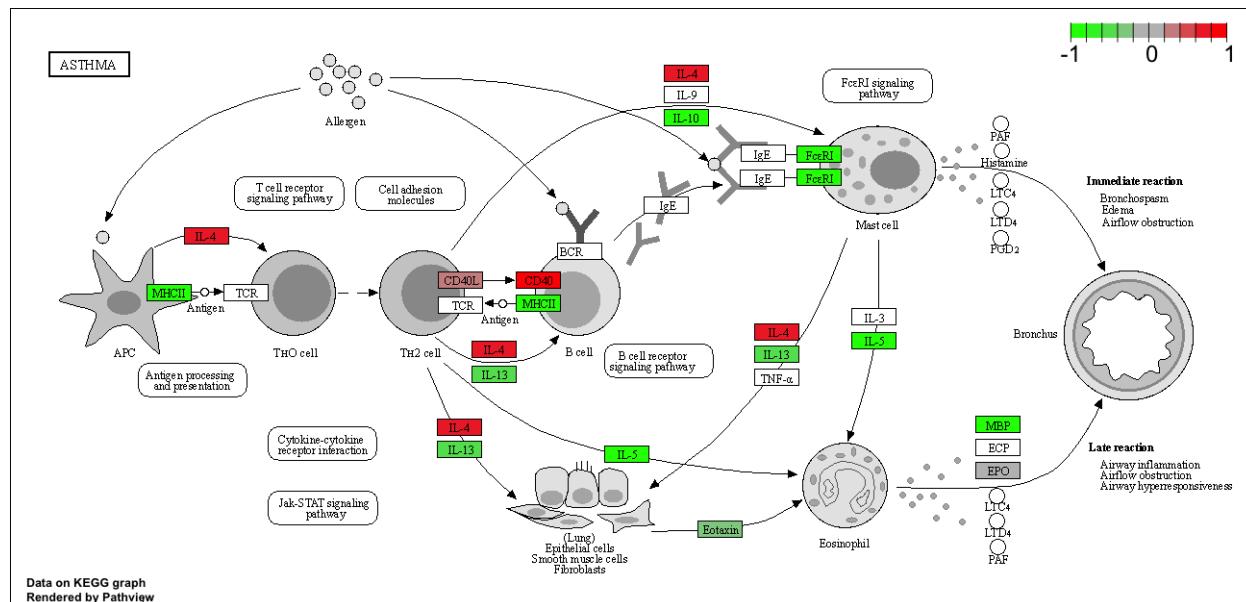
Now I can map my foldchange results onto any KEGG pathway. I will do this manually first by selecting one of the pathway IDs from above.

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

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

```
## Info: Working in directory /Users/kileyhooker/Desktop/BIMM143/week8
```

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



Final step save our results.

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