

Class 12: Transcriptomics and the analysis of RNA-Seq data

Anh Tran

2. Import countData and colData

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

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
```

```
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
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

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

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

```
control treated
        4       4
```

First we should check the corespondence of the metadata and count data

```
metadata$id
```

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

```
colnames(counts)
```

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

To check that these are all in the same order we can use == to test the equality

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

```
[1] TRUE
```

3. Toy differential gene expression

Analysis via comparison of CONTROL vs TREATED

The “treated” have the dex drug and the “control” do not. First I need to be able to extract just the “control” columns in the `metadata` data set

```
control inds <- metadata$dex=="control"  
control <- metadata[control inds,]  
control$id
```

```
[1] "SRR1039508" "SRR1039512" "SRR1039516" "SRR1039520"
```

I can use this to access just the “control” columns of my `counts` data

```
control counts <- counts[,control$id]  
head(control counts)
```

	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

Find the mean count value for each transcript/gene by binding the `rowMeans()`

```
control mean <- rowMeans(control counts)  
head(control mean)
```

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

Q3. How would you make the above code in either approach more robust?

The code `control mean <- rowSums(control counts)/4` is limited because the denominator is fixed for this data set at 4. We can use function `rowMeans` instead to make this approach more robust.

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

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

Now I have control.means and treated.means. Let's put these together for safe keeping and ease of use later.

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

	control.mean	treated.mean
ENSG00000000003	900.75	658.00
ENSG00000000005	0.00	0.00
ENSG00000000419	520.50	546.00
ENSG00000000457	339.75	316.50
ENSG00000000460	97.25	78.75
ENSG00000000938	0.75	0.00

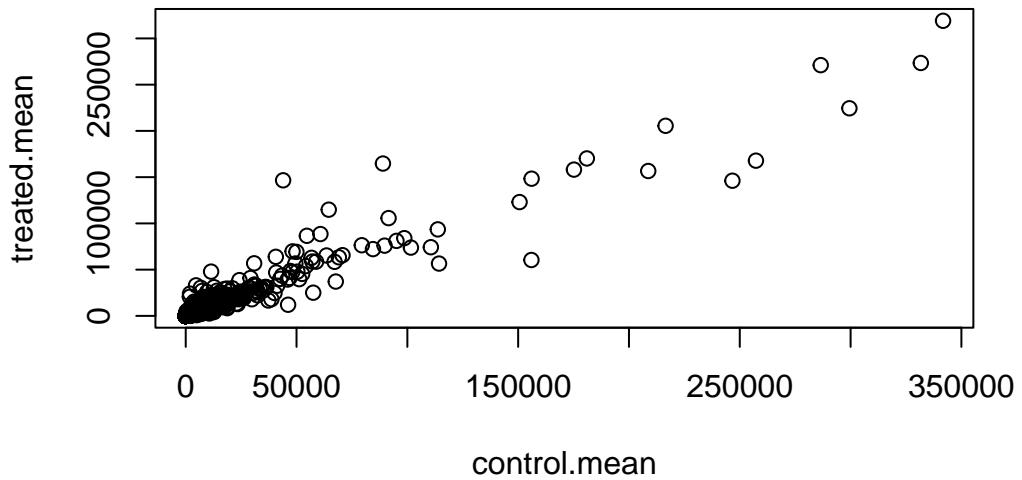
```
colSums(meancounts)
```

	control.mean	treated.mean
	23005324	22196524

Let's do a quick plot to see how our data looks

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.

```
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?

point

This is very heavily skewed and over a wide range - calling out for a log transform

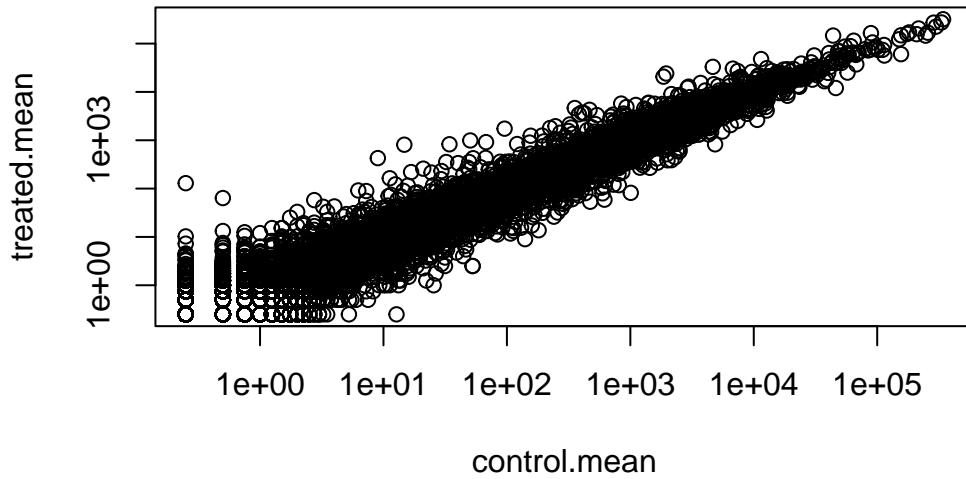
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

The argument is log

```
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



We like working with log transformed data as it can help make things more straightforward to interpret.

If we have no change:

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

```
[1] 0
```

What about if we had a doubling/halving

```
log2(40/20)
```

```
[1] 1
```

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

```
[1] -1
```

We like working with log2 fold-change values. Let's calculate them for our data.

```
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

```
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
ENSG00000001036	2327.00	1785.75	-0.38194109

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 clause 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 dont count any row twice if it has zero entries in both samples.

```
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, these results in their current form are likely to be very misleading. All our analysis has been done based on fold change but fold change can be large 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.

A common threshold for calling genes as differentially expressed is a log2 fold-change of +2 or -2.

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

```
[1] 314
```

What percent is this?

```
round((sum(mycounts$log2fc >= +2)/nrow(mycounts)) * 100)
```

```
[1] 1
```

We need some stats to check if the drug induced difference is significant

Turn to DESeq2

Let's turn to doing this the correct way with the DESeq2 package.

```
library(DESeq2)
```

The main function in the DESeq2 package is called `deseq()`. It wants our count data and our `colData` (metadata) as 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 <- DESeq(dds)
```

estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

```
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

```

```
summary(res)
```

```

out of 25258 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)      : 1563, 6.2%
LFC < 0 (down)    : 1188, 4.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9971, 39%
(mean count < 10)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results

```

```
res05 <- results(dds, alpha=0.05)
summary(res05)
```

```

out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)      : 1236, 4.9%

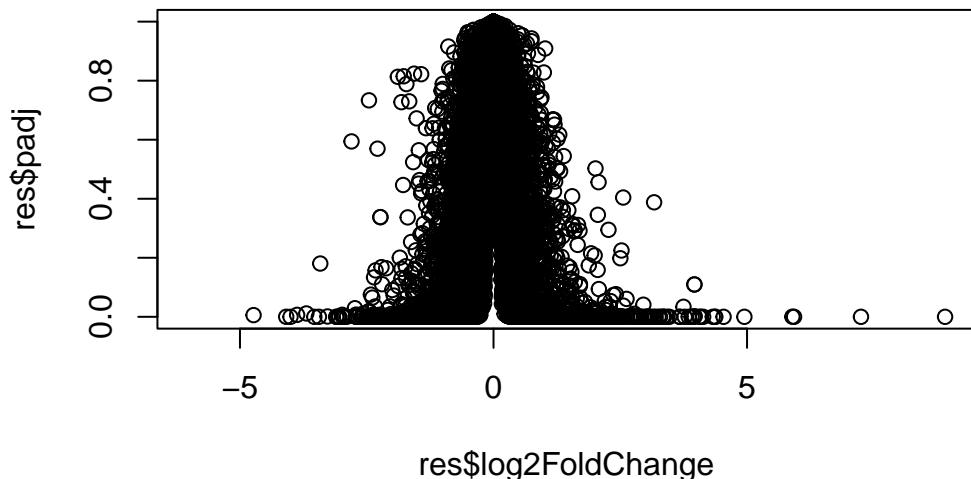
```

```
LFC < 0 (down)      : 933, 3.7%
outliers [1]        : 142, 0.56%
low counts [2]       : 9033, 36%
(mean count < 6)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Adding annotation data

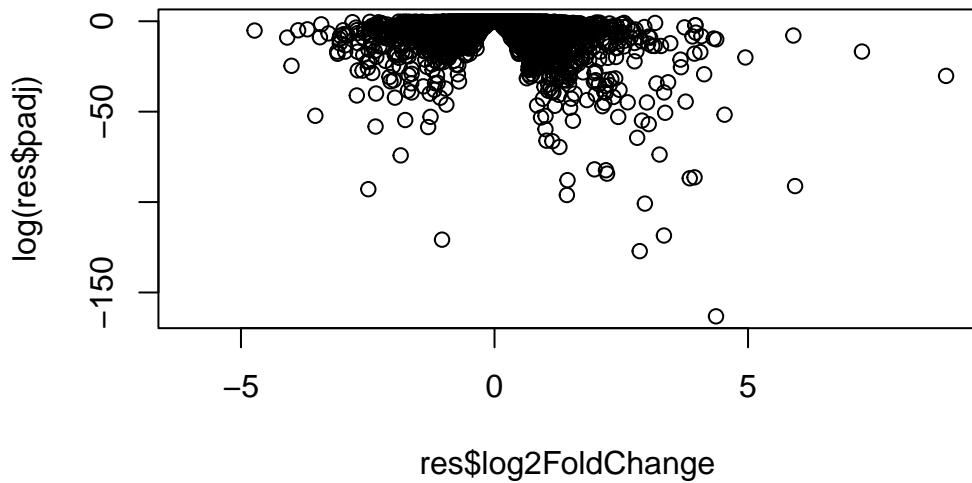
Now what we have got so far is the log2 fold-change and the adjusted p-value for the significance.

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



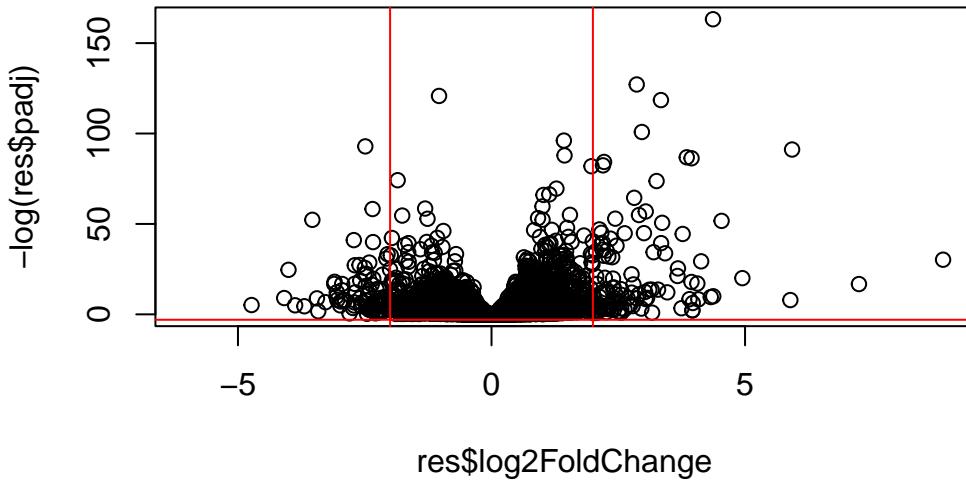
I am going to take the log of the p-value

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



We can flip the y-axis so the plot does not look “upside down”.

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



Annotation of our gene set results

I will start by loading two Annotation packages from bioconductor

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

The `mapID()` function maps database identifiers between different databases. In other words it translates the identifiers used by one database to that used by another database.

Let's see what databases are available for Human data

```
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"
```

My results are in the object `res`

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

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

Q11. Run the `mapIds()` function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called `resentrez`, `resuniprot` 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
```

```

ord <- order( res$padj )
head(res[ord,])

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>
ENSG00000152583    954.771      4.36836  0.2371268   18.4220 8.74490e-76
ENSG00000179094    743.253      2.86389  0.1755693   16.3120 8.10784e-60
ENSG00000116584   2277.913     -1.03470  0.0650984  -15.8944 6.92855e-57
ENSG00000189221   2383.754      3.34154  0.2124058   15.7319 9.14433e-56
ENSG00000120129   3440.704      2.96521  0.2036951   14.5571 5.26424e-48
ENSG00000148175   13493.920     1.42717  0.1003890   14.2164 7.25128e-46
  padj      symbol      entrez      uniprot
  <numeric> <character> <character> <character>
ENSG00000152583 1.32441e-71      SPARCL1      8404  AOA024RDE1
ENSG00000179094 6.13966e-56       PER1        5187  015534
ENSG00000116584 3.49776e-53      ARHGEF2      9181  Q92974
ENSG00000189221 3.46227e-52       MAOA        4128  P21397
ENSG00000120129 1.59454e-44      DUSP1        1843  B4DU40
ENSG00000148175 1.83034e-42       STOM        2040  F8VSL7
  genename
  <character>
ENSG00000152583          SPARC like 1
ENSG00000179094          period circadian reg..
ENSG00000116584          Rho/Rac guanine nucl..
ENSG00000189221          monoamine oxidase A
ENSG00000120129          dual specificity pho..
ENSG00000148175          stomatin

write.csv(res[ord,], "deseq_results.csv")

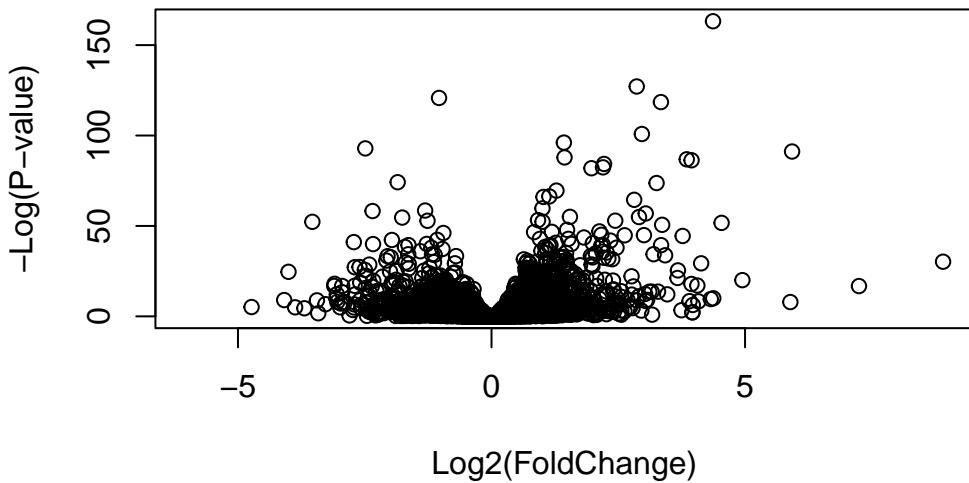
```

Data Visualization

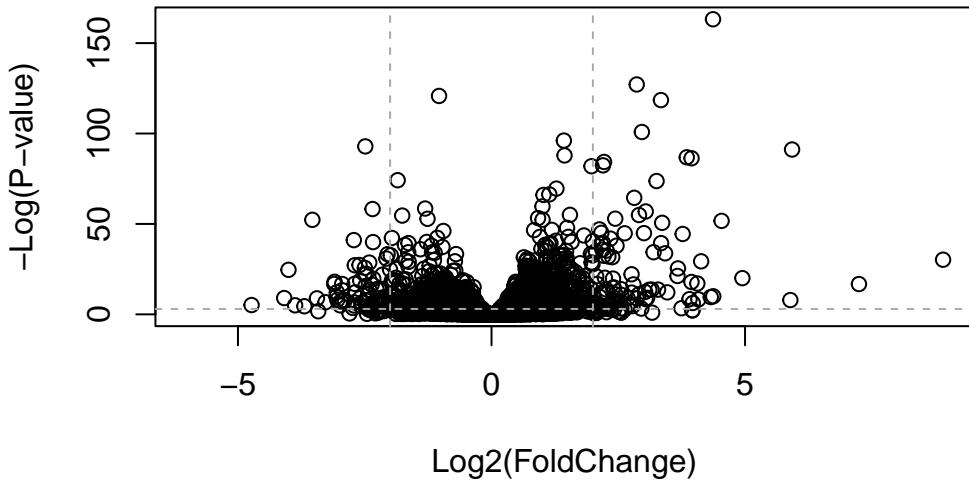
```

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

```



```
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)
```



```

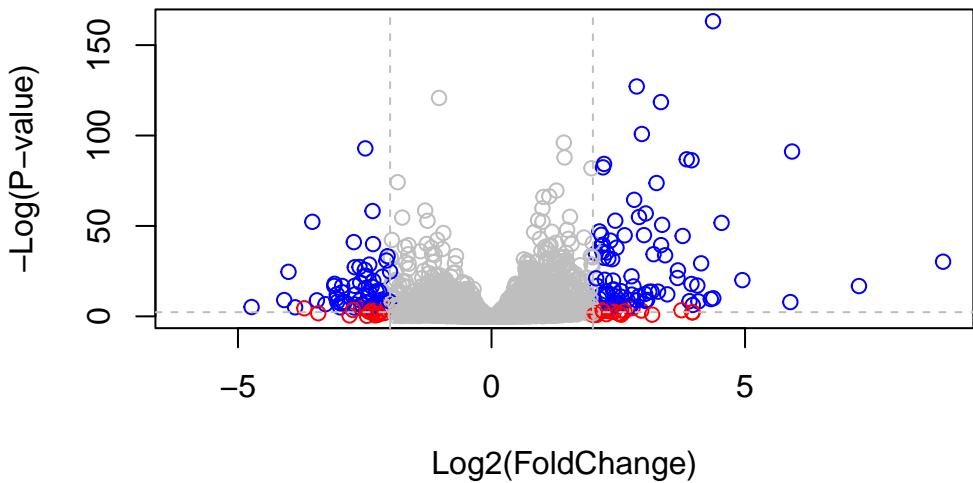
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

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)

```



```
library(EnhancedVolcano)
```

```
Loading required package: ggplot2
```

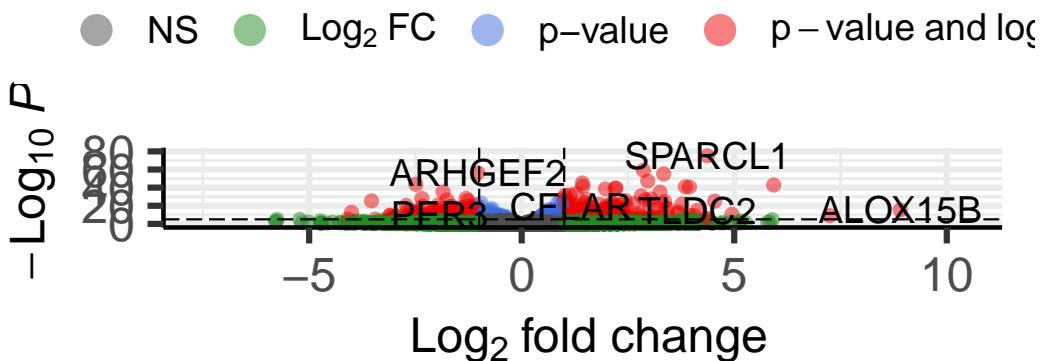
```
Loading required package: ggrepel
```

```
x <- as.data.frame(res)

EnhancedVolcano(x,
  lab = x$symbol,
  x = 'log2FoldChange',
  y = 'pvalue')
```

Volcano plot

Enhanced Volcano



total = 38694 variables

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)

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"

```

The main `gage()` function wants a vector as input that contains our measure of importance - in our case that is fold-change. The vector needs to have ENTREZ ids as the names of the vector.

Recall that vectors have names - this is useful for book-keeping so I know what value corresponds to a certain gene for example.

```

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

Now we can run the analysis

```

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

```

What is in this results object

```

attributes(keggres)

```

```
$names  
[1] "greater" "less"     "stats"
```

By default gage splits it's results into “greater” and “less” objects that you can examine

```
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

We can now look in more detail at these pathways. The `pathview()` function will take the KEGG pathway ID (printed first above) and our vector of importance and annotate the pathway with our guests.

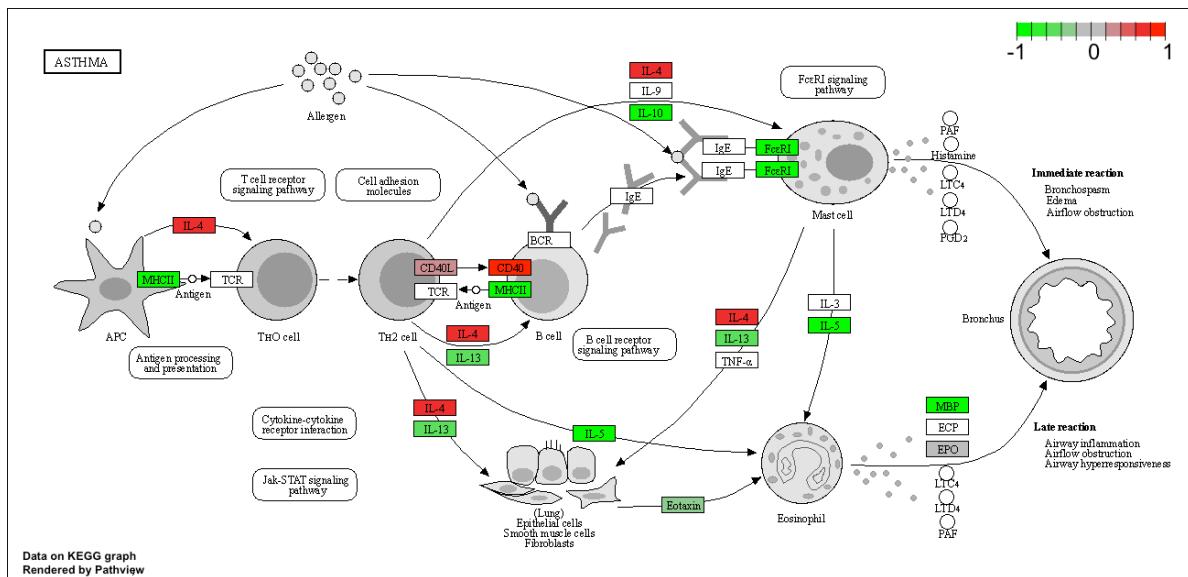
First I will look at hsa05310 Asthma

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

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

```
Info: Working in directory /Users/anhtran/Documents/UCSD/FA22/BIMM 143 - Introduction to Bio
```

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



A different PDF based output of the same data

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

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

```
Info: Working in directory /Users/anhtran/Documents/UCSD/FA22/BIMM 143 - Introduction to Bio
```

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

Q12. Can you do the same procedure as above to plot the pathview figures for the top 2 down-regulated pathways?

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

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

```
Info: Working in directory /Users/anhtran/Documents/UCSD/FA22/BIMM 143 - Introduction to Bio
```

```
Info: Writing image file hsa05332.pathview.png
```

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

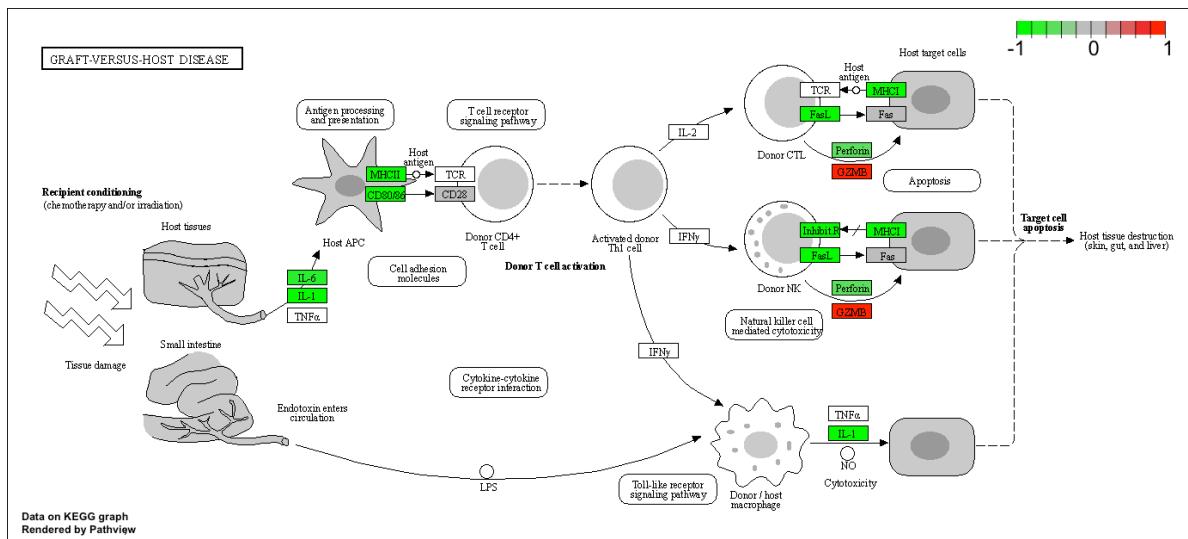


Figure 1: The Graft-versus-host disease pathway

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

```
Info: Working in directory /Users/anhtran/Documents/UCSD/FA22/BIMM 143 - Introduction to Bio
```

```
Info: Writing image file hsa04940.pathview.png
```

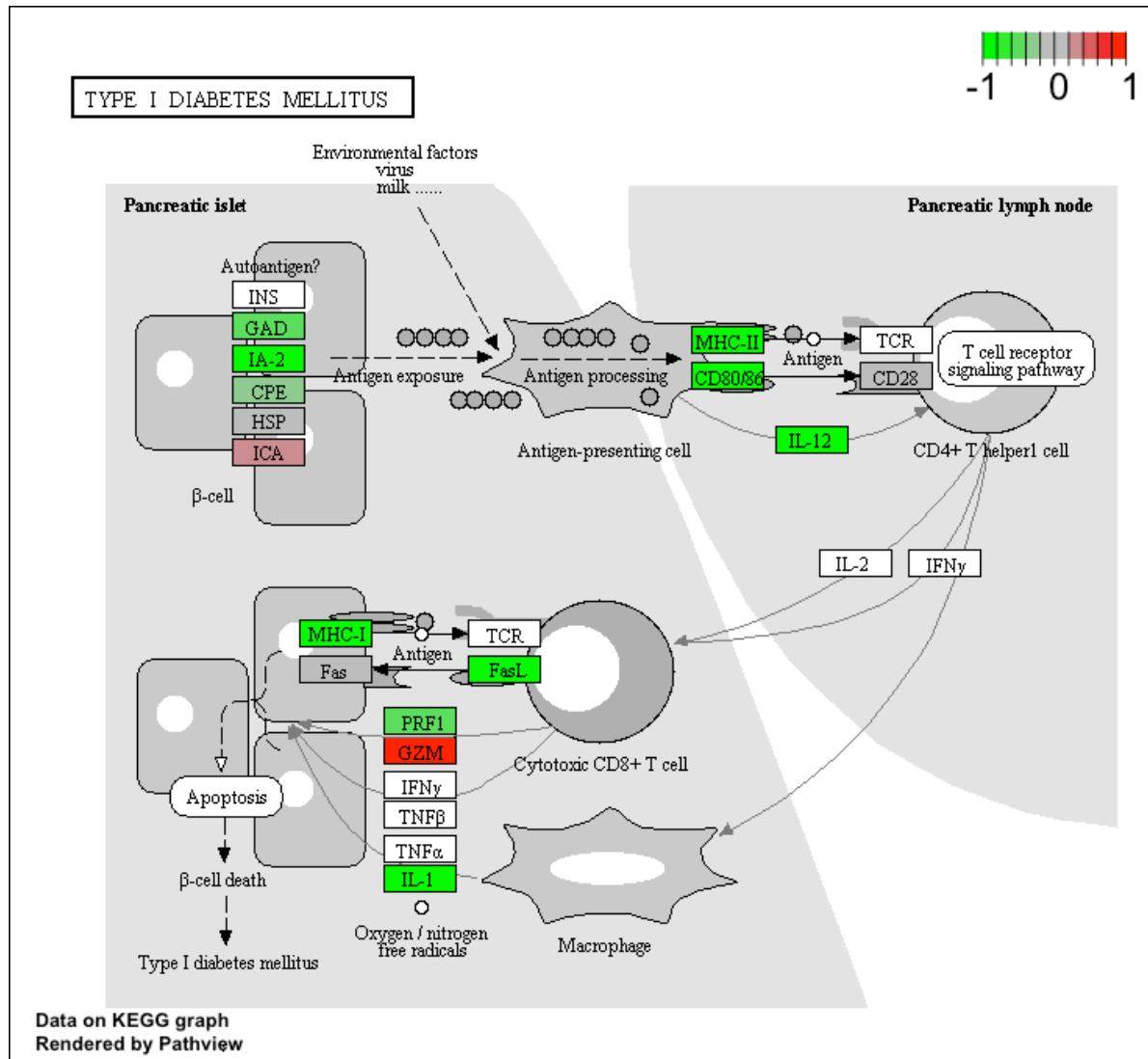


Figure 2: Type I diabetes mellitus