Class 13: Transcriptomics and the analysis of RNA-Seq data

Kris Price (PID: A17464127)

Table of contents

Background

Today we will perform an RNASeq analysis of the effects of a common steroid on airway cells.

In particular, dexamethasone (hereafter, just called "dex") on different airway smooth muscle cell lines (ASM cells).

Data Import

We need two different inputs:

- **countData**: with genes in rows and experiments in columns
- **colData**: metadata that describes the columns in countData

```
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
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38,694 genes in this dataset.

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

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

```
control treated
      4       4
```

```
# or we could do sum(metadata$dex == "control")
```

There's 4 'control' cell lines.

Differential Gene Expression

We have 4 replicate drug treated and control (no drug) columns/experiments in our `counts` object.

We want one "mean" value for each gene (rows) in "treated" (drug) and one mean value for each gene in "control" cols.

Step 1. Find all "control" columns in `counts` Step 2. Extract these columns to a new object called `control.counts` Step 3. Then calculate the mean value for each gene

```
library(tidyverse)
```

```
control.counts <- metadata %>%
  filter(dex == "control") %>%
  pull(id)

control.mean <- counts %>%
  select(control.counts) %>%
  rowMeans()
```

```
Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
i Please use `all_of()` or `any_of()` instead.
  # Was:
  data %>% select(control.counts)

  # Now:
  data %>% select(all_of(control.counts))

See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```
head(control.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

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

Instead of defining `control.means` with `rowSums(control.counts)/4`, you could just directly calculate the mean using `rowMeans(control.counts)`.

> 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)

Now do the same thing for the "treated" columns/experiments...

```
treated.counts <- metadata %>%
  filter(dex == "treated") %>%
  pull(id)

treated.mean <- counts %>%
  select(treated.counts) %>%
  rowMeans()
```

```
Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
i Please use `all_of()` or `any_of()` instead.
  # Was:
  data %>% select(treated.counts)

  # Now:
  data %>% select(all_of(treated.counts))

See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

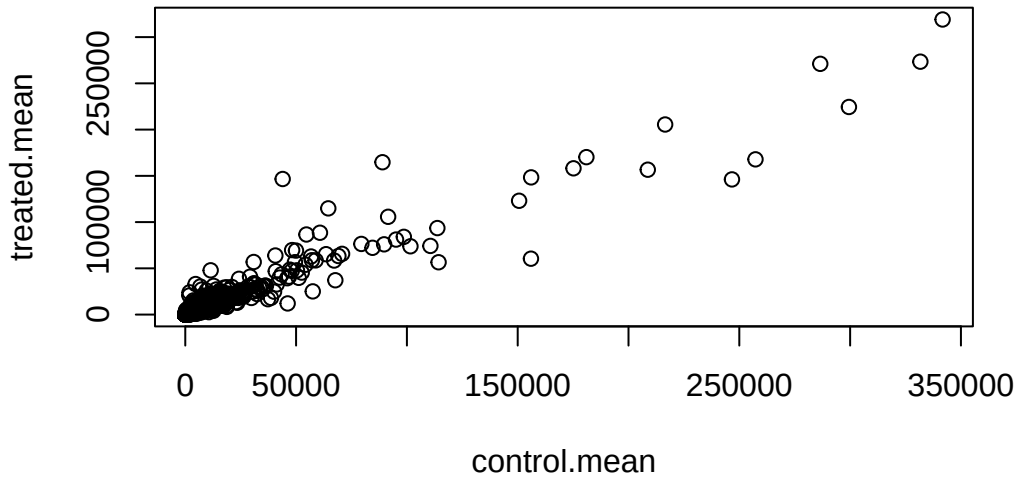
```
head(treated.mean)
```

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

Q5. 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.

Put these together for easy book-keeping as a `mean.counts` dataframe. Then, create a quick plot (note that, when using ggplot, you can create this same plot using `geom_point()`:

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

plot(mean.counts)
```



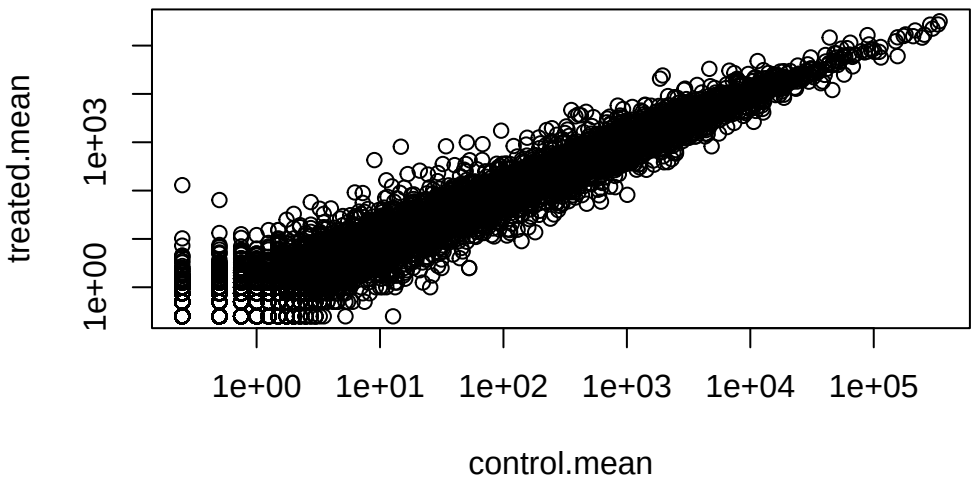
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

Let's log transform this count data, using the `log =` argument:

```
plot(mean.counts, 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
```



N.B. We most often use log2 for this type of data as it makes the interpretation much more straightforward.

Treated/Control is often called "fold-change"

If there was no change, we would have a log2-fc of 0:

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

```
[1] 0
```

If we had double the amount of transcript around, we'd have a log2-fc of 1:

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

```
[1] 1
```

If we had half as much transcript around, we'd have a log2-fc of -1:

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

```
[1] -1
```

Q. Calculate a log2 fold change value for all our genes and add it as a new column to our `mean.counts` object.

```
mean.counts <- mean.counts %>%
  mutate(log.fc = log2(treated.mean / control.mean))

head(mean.counts)
```

```
                control.mean treated.mean      log.fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000005         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
```

There's some "funky" log2fc values (NaN and -Inf) here that come about whenever we have 0 mean count values. Typically we would remove these genes from any further analysis, as we can't say anything about them if we have no data for them.

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

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

```
                control.mean treated.mean      log.fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000971      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` argument causes the `which()` function to return the array indices (AKA both the row and column indices) where the mean count is 0. The `unique()` function ensures that each row can only be counted up to one time, even if it has a mean of 0 in both experiments.

```
up.ind <- mycounts$log.fc > 2
down.ind <- mycounts$log.fc < (-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
```

There are 250 up-regulated genes that are greater than 2 fc level.

> 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
```

There are 367 down-regulated genes that are greater than 2 fc level.

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

I would not readily trust these results because even though these genes show such a large difference in expression across treatments, there are no measurements of statistical significance.

DESeq Analysis

Let's do this analysis with an estimate of statistical significance using the **DESEq2** package.

```
library(DESeq2)
```

DESeq2 (like many BioConductor packages) wants its input data in a very 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
```

Run the DESeq analysis pipeline

The main function `DESeq()`

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

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

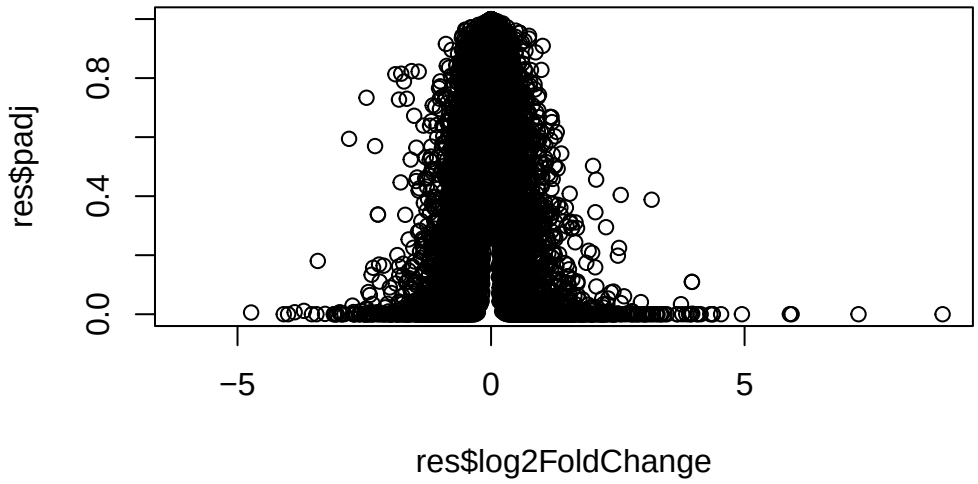
```
res <- results(dds)

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>
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
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

Volcano Plot

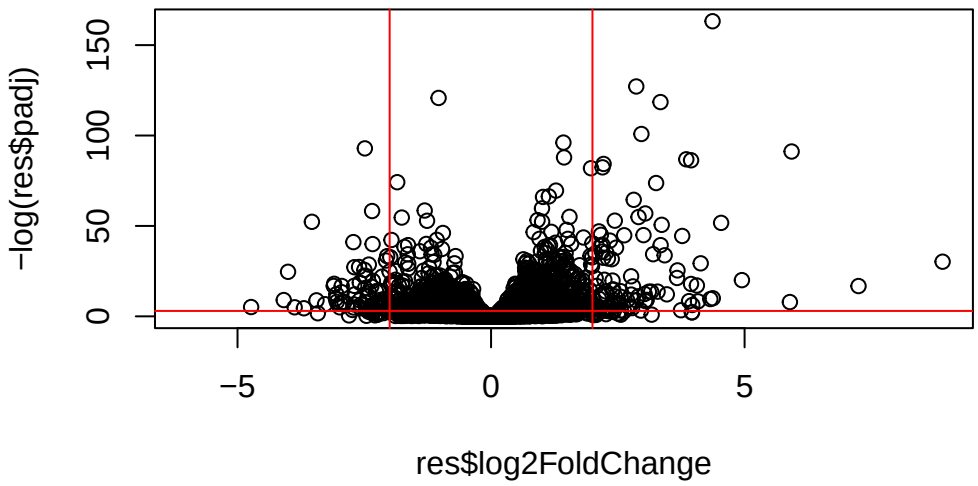
This is a main summary results figure from these kinds of studies. It's a plot of log2 fold-change vs. adjusted p-value.

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



Again, this y-axis is highly skewed, and needs log transforming. We can flip the y-axis with a minus sign so it looks like every other volcano plot.

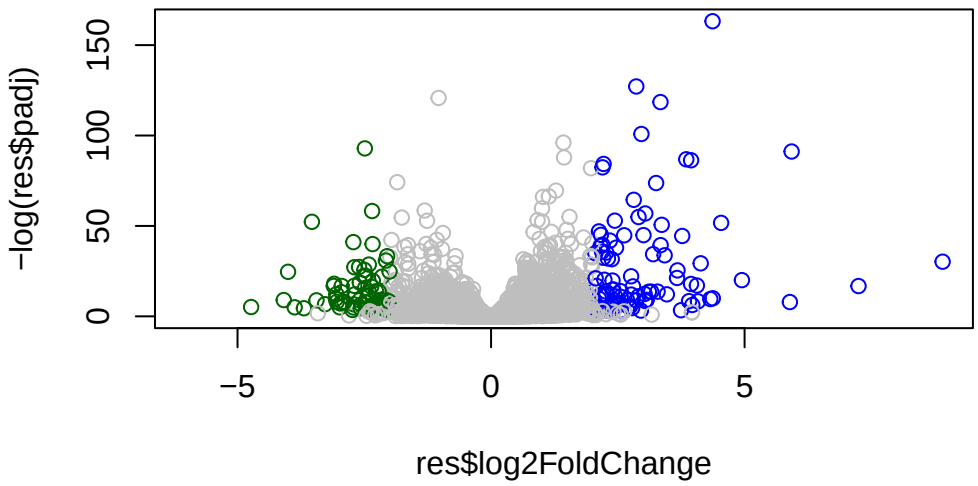
```
plot(res$log2FoldChange, -log(res$padj))

abline(v = -2, col = "red")
abline(v = 2, col = "red")
abline(h = -log(0.05), col = "red")
```



Adding some color annotation

Start with a default base color "gray"

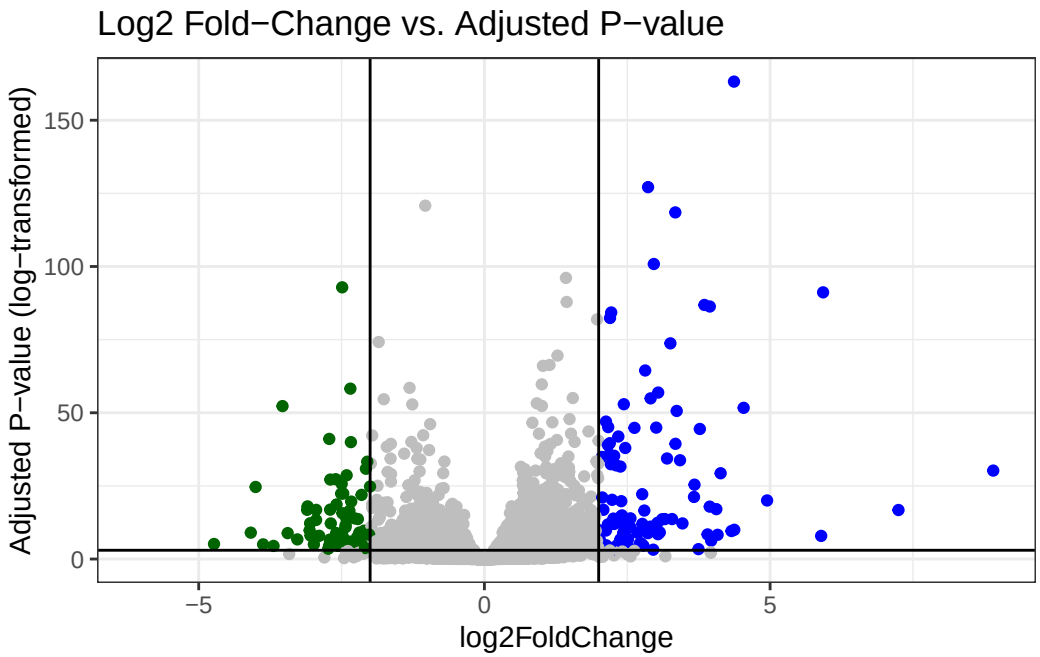
```
mycols <- rep("gray", nrow(res))

mycols[res$log2FoldChange > 2] <- "blue"
mycols[res$log2FoldChange < -2] <- "darkgreen"
mycols[res$padj >= 0.05] <- "gray"

plot(res$log2FoldChange, -log(res$padj), col = mycols)
```



Q. Make a presentation- quality ggplot version of this plot. Include clear axis labels, a clean theme, your custom colors, cut-off lines and a plot title.

```
ggplot(res, aes(log2FoldChange, -log(padj))) +
  geom_point(col = mycols) +
  geom_vline(aes(xintercept = -2)) +
  geom_vline(aes(xintercept = 2)) +
  geom_hline(aes(yintercept = -log(0.05))) +
  labs(title = "Log2 Fold-Change vs. Adjusted P-value") +
  ylab("Adjusted P-value (log-transformed)") +
  theme_bw()
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Save our results

Write a CSV file:

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

Add annotation data

We need to add missing annotation data to our main `res` results object. This includes the common gene "symbol"

```
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>
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
                  <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

We will use R and bioconductor to do this "ID mapping"

```
library(AnnotationDbi)
```

```
Attaching package: 'AnnotationDbi'

The following object is masked from 'package:dplyr':

    select
```

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

Let's see what databases we can use for translation/mapping...

```
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"
```

We can use the `mapIds()` function now to "translate" between any of these databases.

```
res$symbol <- mapIds(org.Hs.eg.db,
                     keys = row.names(res), # Our gene names
                     keytype = "ENSEMBL", # The format of our gene names
                     column = "SYMBOL") # The new format we want to add
```

```
'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>
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
                <numeric> <character>
ENSG00000000003  0.163035      TSPAN6
ENSG00000000005        NA        TNMD
ENSG00000000419  0.176032        DPM1
ENSG00000000457  0.961694       SCYL3
ENSG00000000460  0.815849       FIRRM
ENSG00000000938        NA         FGR
```

Q. Also add "ENTReZID", "GENENAME" IDs to our `res` object:

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys = row.names(res),
                     keytype = "ENSEMBL",
                     column = "ENTREZID")
```

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

```
res$genename <- mapIds(org.Hs.eg.db,
                     keys = row.names(res),
                     keytype = "ENSEMBL",
                     column = "GENENAME")
```

```
'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>
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              genename
                <numeric> <character> <character>           <character>
ENSG00000000003  0.163035      TSPAN6        7105          tetraspanin 6
ENSG00000000005        NA        TNMD       64102            tenomodulin
ENSG00000000419  0.176032        DPM1        8813 dolichyl-phosphate m..
ENSG00000000457  0.961694       SCYL3       57147 SCY1 like pseudokina..
ENSG00000000460  0.815849       FIRRM       55732 FIGNL1 interacting r..
ENSG00000000938        NA         FGR        2268 FGR proto-oncogene, ..
```

Save annotated results to a CSV file

```
write.csv(res, "results_annotated.csv")
```

Pathway Analysis

What known biological pathways do our differentially expressed genes overlap with (i.e. play a role in)?

There's lots of bioconductor packages to do this type of analysis.

We will use one of the oldest, called **gage**, along with **pathview** to render nice pictures of the pathways we find.

```
library(pathview)
library(gage)
library(gageData)
```

Have a little peak into what's in 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 that does the work wants a simple vector as input.

```
foldchanges <- res$log2FoldChange
names(foldchanges) <- res$entrez # The KEGG database uses ENTREZ IDs, so we need to provide t

head(foldchanges)
```

```
       7105       64102        8813       57147       55732        2268
-0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

Now we can run gage()

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

What's in the output object keggres?

```
attributes(keggres)
```

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

```
# Look in 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
```

We can use the **pathview** function to render a figure of any of these pathways along with annotation for our DEGs.

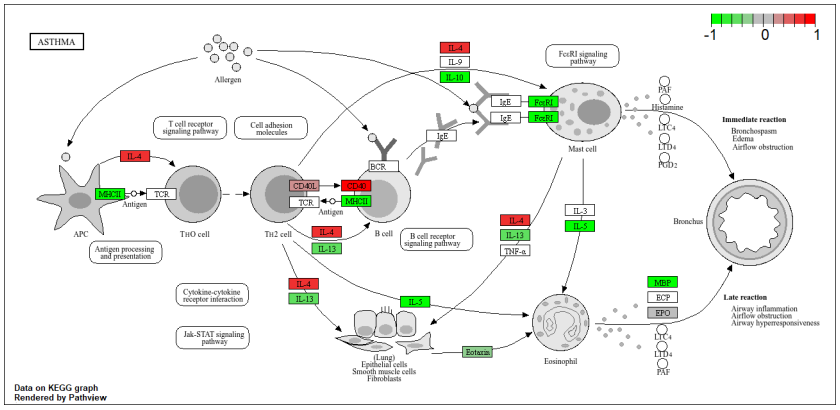
Let's see the hsa05310 Asthma pathway with our DEGs colored up:

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

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

```
Info: Working in directory C:/Users/kryan/Desktop/School/BIMM 143/class13
```

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



Q. Can you render and insert here the pathway figure for "Graft-versus-host disease" and "Type I diabetes?"

Graft-versus-host disease:

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

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

```
Info: Working in directory C:/Users/kryan/Desktop/School/BIMM 143/class13
```

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

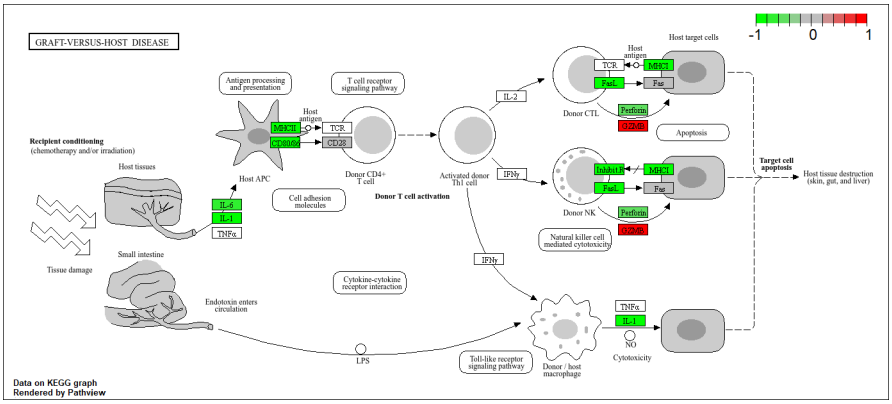


Figure 1: Graft-versus-host disease pathview

Type I diabetes:

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

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

```
Info: Working in directory C:/Users/kryan/Desktop/School/BIMM 143/class13
```

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

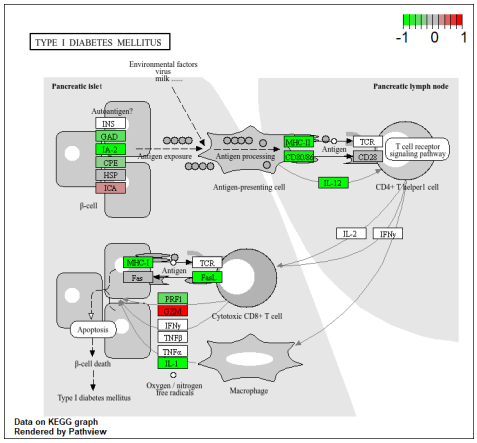


Figure 2: Type I diabetes pathview