Transcriptomics and the analysis of RNA-Seq data

Duc Nguyen

1. Bioconductor and DESeq2 setup

Note: Install Bioconductor packages by using 'install.packages("BiocManager")' and 'BiocManager::install()' functions. Install the DESeq2 bioconductor package by using 'BiocManager::install("DESeq2")' function.

```
library(BiocManager)
library(DESeq2)
```

2. Import countData and colData

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

```
metadata <-  read.csv("airway_metadata.csv")
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
```

Ans1: 38694 genes are in this dataset

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

```
ncol(metadata)
```

```
[1] 4
```

Ans2: There are 4 'control' cell lines

3. Toy differential gene expression

Filter "control" samples in "dex" column

```
control <- metadata[metadata[,"dex"] == "control",]
head(control)
```

```
          id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
3 SRR1039512 control  N052611 GSM1275866
5 SRR1039516 control  N080611 GSM1275870
7 SRR1039520 control  N061011 GSM1275874
```

Calculate the mean per gene across control samples

```
control.counts <- counts[,control$id]
control.mean <- rowSums(control.counts)/4
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?

Ans3: If we add more samples, the values obtained with the exact code above will not be correct anymore because the control.mean is calculated by dividing by 4 samples; thus, if adding more samples, the output value will be wrong. For making the code more robust, instead of using the 'rowSums()' function then dividing by 4 in the code control.mean <- rowSums(control.counts)/4, we can use the the 'rowMeans()' function like control.mean <- 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)

Ans4:

```
treated <- metadata[metadata[,"dex"] == "treated",]
head(treated)
```

```
          id     dex celltype     geo_id
2 SRR1039509 treated   N61311 GSM1275863
4 SRR1039513 treated  N052611 GSM1275867
6 SRR1039517 treated  N080611 GSM1275871
8 SRR1039521 treated  N061011 GSM1275875
```

```
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
```

Combine 'control.mean' and 'treated.mean' into 'meancounts' data.frame

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

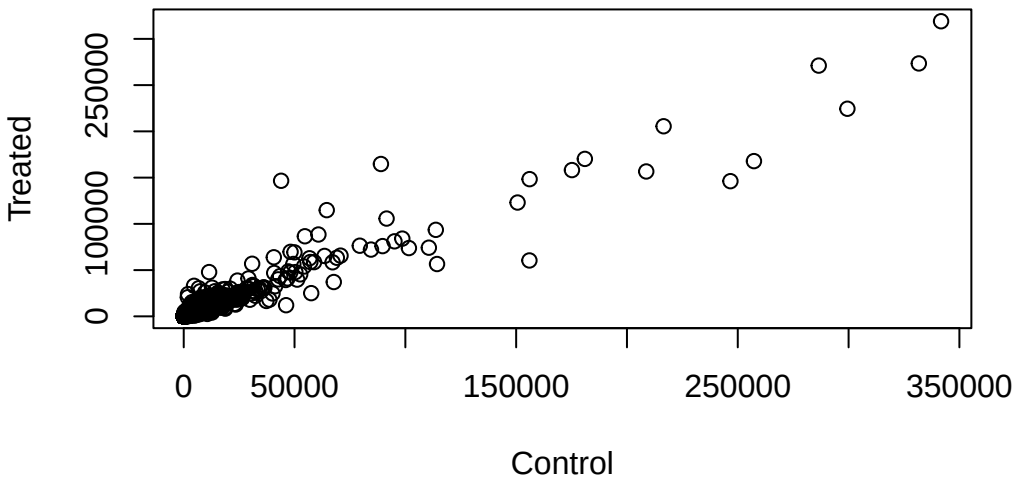
The sum of the mean counts across all genes for each group

```
colSums(meancounts)
```

```
control.mean treated.mean
    23005324     22196524
```

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.

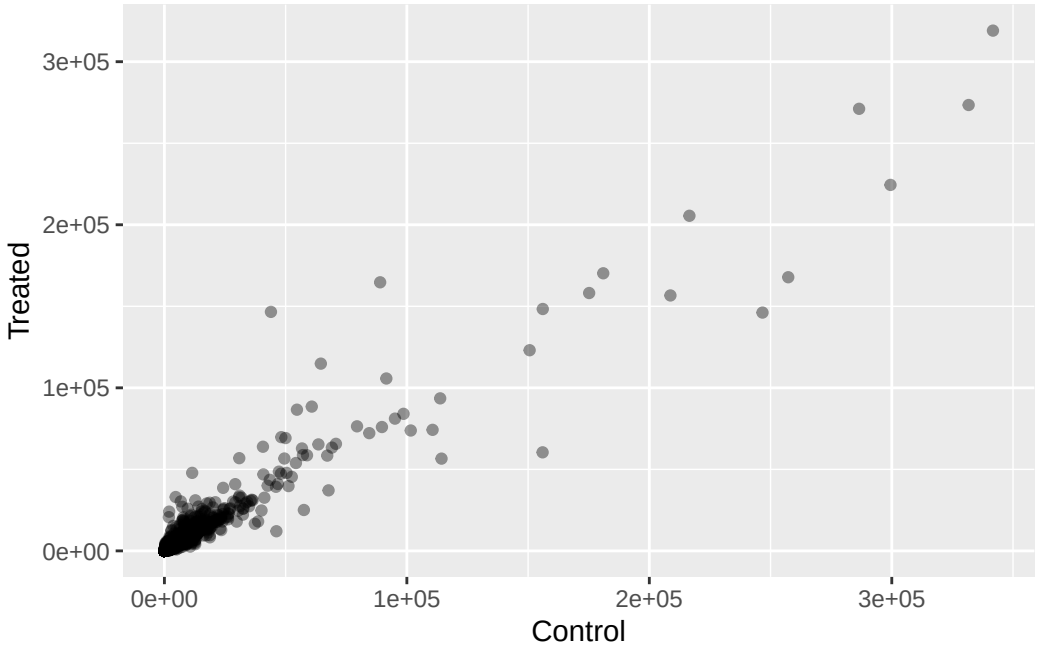
```
plot(x = meancounts$control.mean, y = meancounts$treated.mean,
     xlab = "Control", ylab = "Treated")
```



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?

Ans5: point

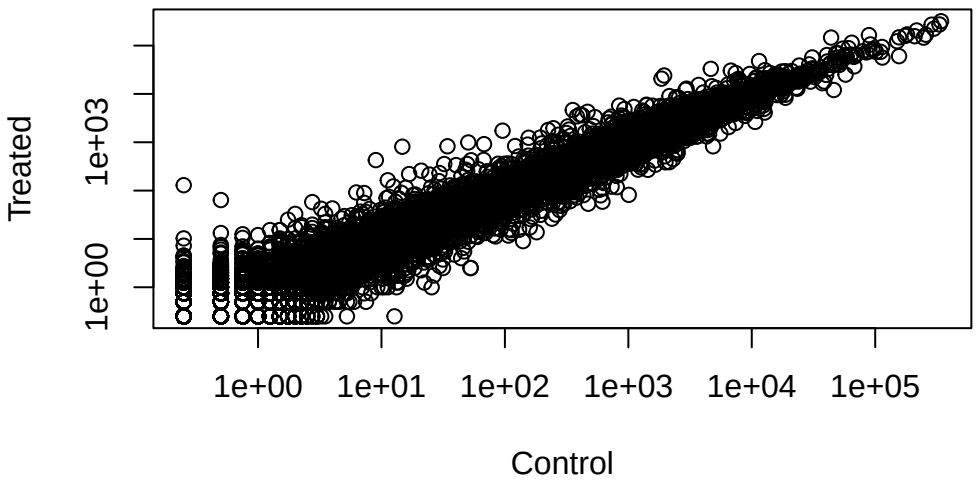
```
library(ggplot2)

ggplot(meancounts) +
  aes(x = control.mean, y = treated.mean) +
  geom_point(alpha = 0.4) +
  labs(x = "Control", y = "Treated")
```



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

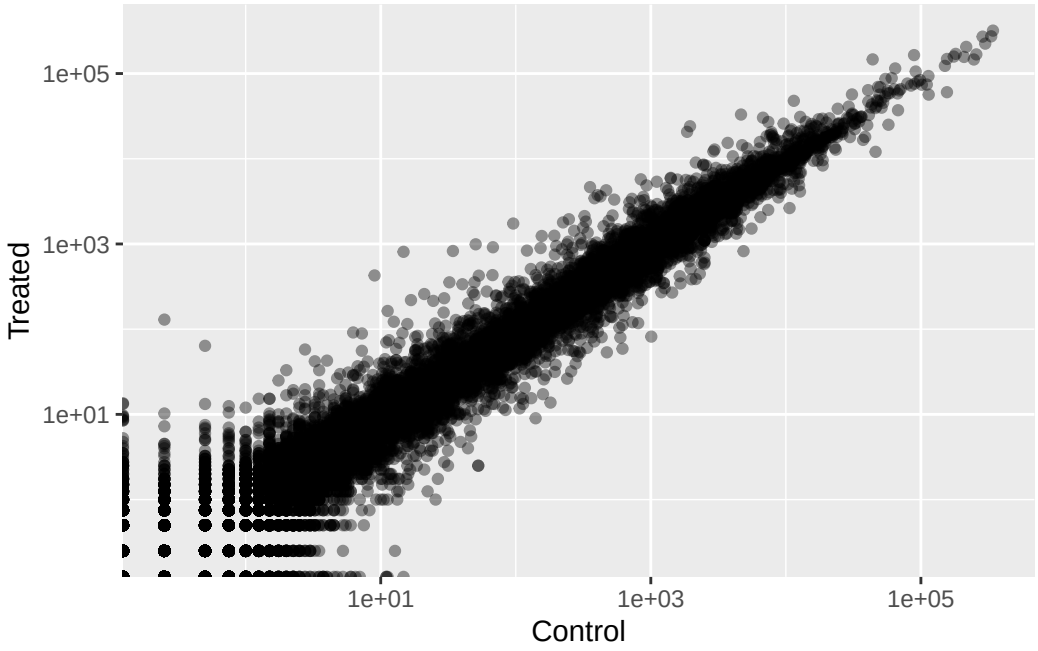
Ans6: log

```
plot(x = meancounts$control.mean, y = meancounts$treated.mean,
     log = "xy", xlab = "Control", ylab = "Treated")
```



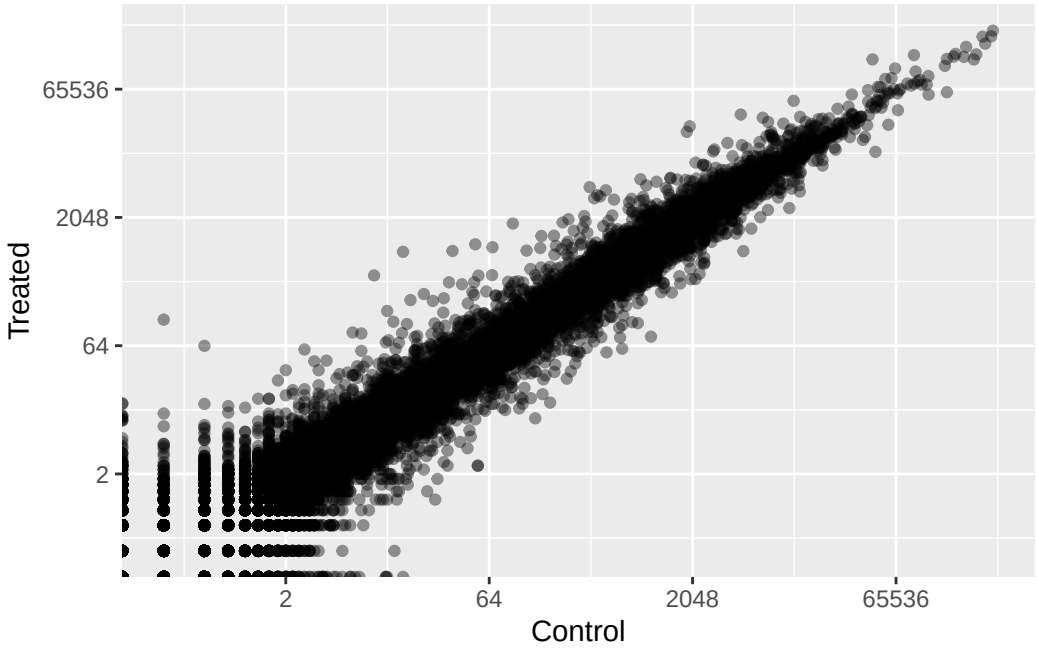
Using ggplot2 with log10:

```
ggplot(meancounts) +
  aes(x = control.mean, y = treated.mean) +
  geom_point(alpha = 0.4) +
  labs(x = "Control", y = "Treated") +
  scale_x_log10() +
  scale_y_log10()
```



Using ggplot2 with log2:

```
ggplot(meancounts) +
  aes(x = control.mean, y = treated.mean) +
  geom_point(alpha = 0.4) +
  labs(x = "Control", y = "Treated") +
  scale_x_continuous(trans ="log2") +
  scale_y_continuous(trans = "log2")
```



Calculate log2foldchange and then add it to the 'meancounts' data.frame

```
meancounts$log2fc <- log2(meancounts[,"treated.mean"]/meancounts[,"control.mean"])
head(meancounts)
```

```
                control.mean treated.mean      log2fc
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
```

Filter out any rows that have the NaN ("not a number") and -Inf (negative infinity) results in the log2fc column

```
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
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?

Ans7: The 'arr.ind = TRUE' argument in the 'which()' function above is used to return the positions where the TRUE values are by pointing to the row and column indices of the TRUE values, in this case, it is used to point out which genes (rows) and samples (columns) have zero counts. The 'unique()' function is used to ensure that does not count any row twice if it has zero entries in both samples.

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

```
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
```

Ans8: 250 up regulated genes

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
```

Ans9: 367 down regulated genes

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

Ans10: We cannot trust these results right away since we did not do anything so far to determine whether the differences have statistically significant or not. We need to perform statistics and calculate their p-values to see if their p-values < 0.05 for statistically significant.

4. DESeq2 analysis

```
library(DESeq2)
citation("DESeq2")
```

```
To cite package 'DESeq2' in publications use:

  Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change
  and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550
  (2014)

A BibTeX entry for LaTeX users is

  @Article{,
    title = {Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2
    author = {Michael I. Love and Wolfgang Huber and Simon Anders},
    year = {2014},
    journal = {Genome Biology},
    doi = {10.1186/s13059-014-0550-8},
    volume = {15},
    issue = {12},
    pages = {550},
  }
```

Importing data

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)
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
```

DESeq analysis and getting results

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

Using the summary function to summarize some basic tallies

```
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
```

Adjust the p-value cutoff < 0.05 for statistic significant

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

```
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
```

5. Adding annotation data (Skipped)

Go back and start working on 11/8/2022

Install the AnnotationDbi package by using the 'BiocManager::install("AnnotationDbi")' function and the annotation data package for humans org.Hs.eg.db by using the 'BiocManager::install("org.Hs.eg.db")' function.

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

The 'mapIDs()' func. "maps" database identifiers between different database. In other words, it translates the identifiers used by one database to that used by other database.

Let's see what database are available for Human data

```
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"
```

My results are in the object "res"

```
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
```

```
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")
```

```
head(res$symbol)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
       "TSPAN6"          "TNMD"          "DPM1"         "SCYL3"      "C1orf112"
ENSG00000000938
          "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$uniprot and res$genenam

Ans11

Add the Entrez ID

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

Add the UniProt accession

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

Add the GENENAME

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

```
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  A0A024RCI0
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.176032        DPM1        8813      O60762
ENSG00000000457  0.961694       SCYL3       57147      Q8IZE3
ENSG00000000460  0.815849    C1orf112       55732  A0A024R922
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, ..
```

```
ord <- order(res$padj)
View(res[ord,])
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  A0A024RDE1
ENSG00000179094 6.13966e-56        PER1        5187      O15534
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
```

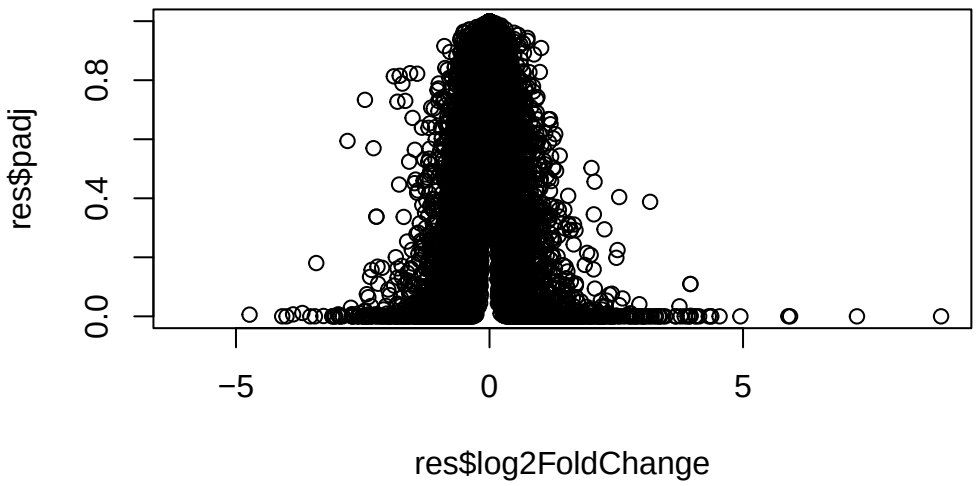
Finally, let's write out the ordered significant results with annotations

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

6. Data Visualization

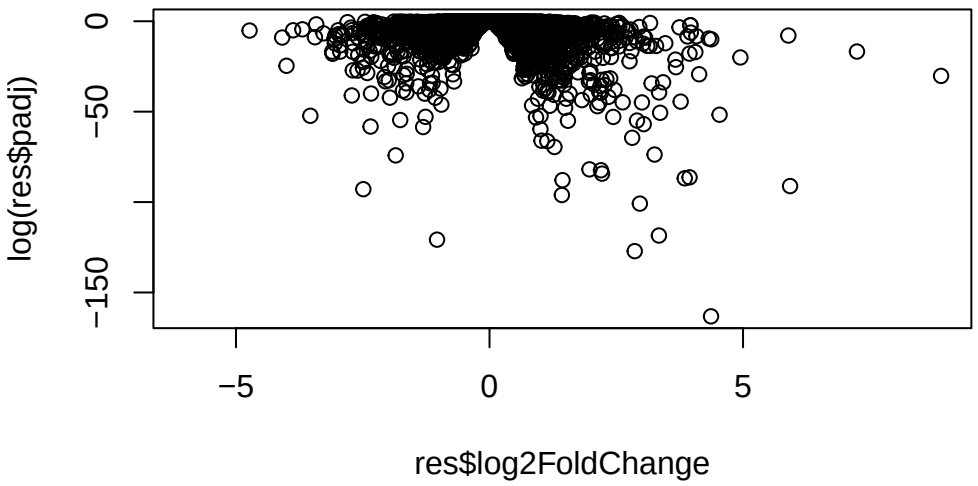
A first plot

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



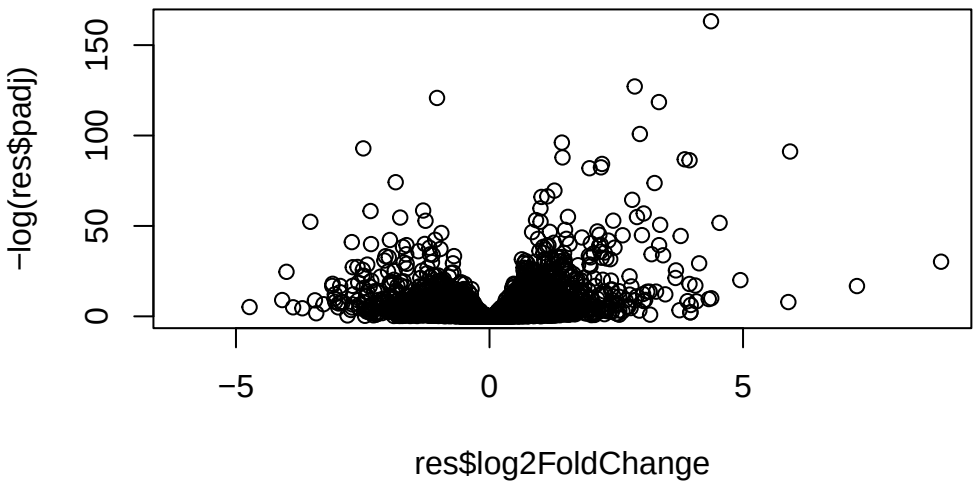
Take the log of the p-values

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



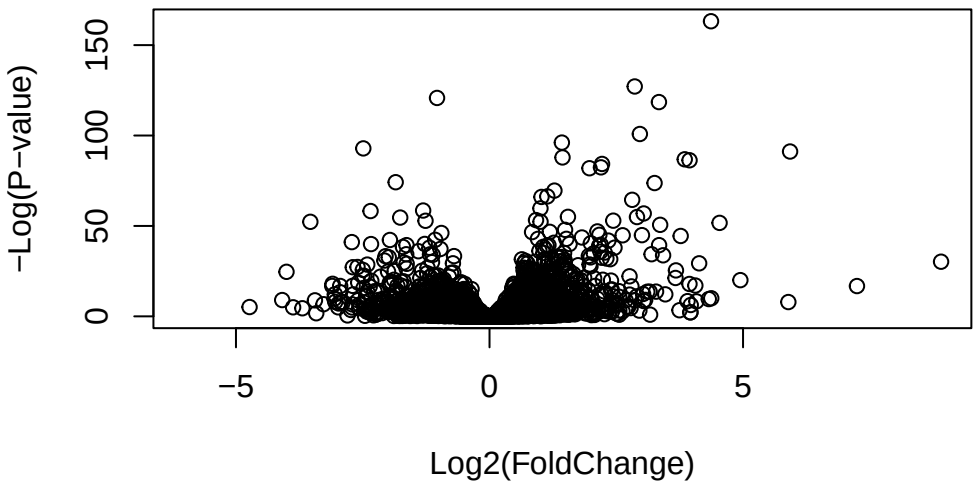
Flip the y-axis so the plot does not look "upside down"

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



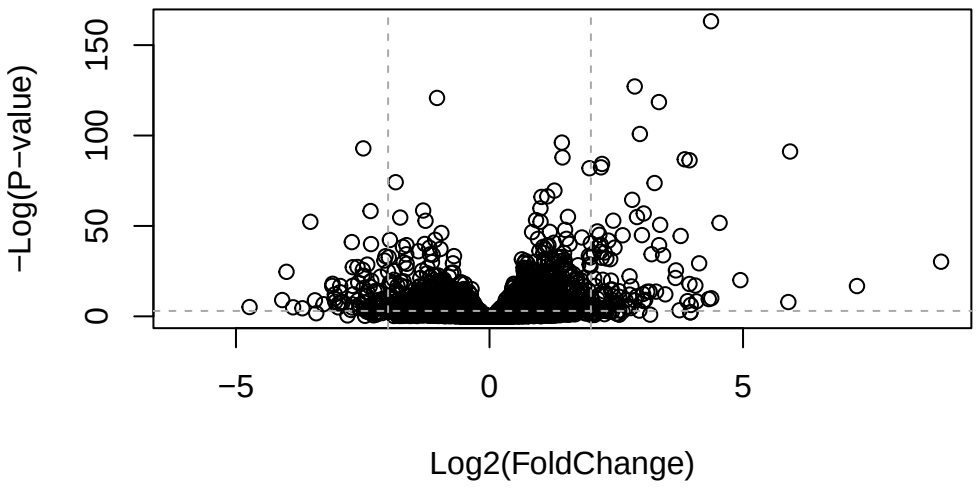
Volcano plots

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



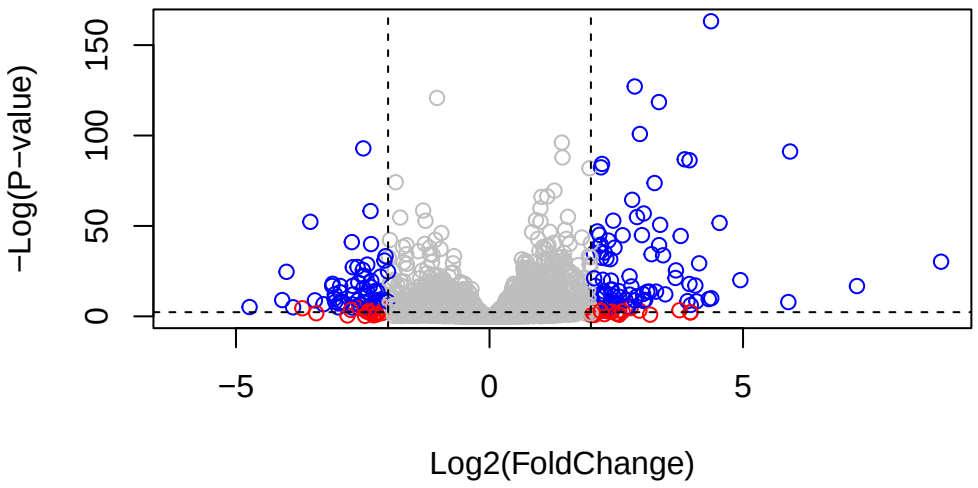
Add some guidelines (with the 'abline()' function) and color (with a custom color vector) highlighting genes that have padj < 0.05 and the absolute log2FoldChange > 2

```
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)
```



Customize color vector indicating transcripts with large fold change and significant differences between conditions

```
# 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 = "black", lty = 2)
abline(h = -log(0.1), col = "black", lty = 2)
```



7. Pathway analysis (skipped)

Go back and start working on 11/8/2022

Some major genesets include KEGG, GO, etc.

Install the pathview package by using the 'BiocManager::install(c("pathview", "gage", "gageData"))' function

We will use the 'gage' package for our first pathway analysis

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

data(kegg.sets.hs)
```

We have to look at the first few pathways the the kegg human set

```
# 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"
```

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

```
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 its results into "greater" and "less" objects that you can examine. First, we look at the "less" (i.e.down regulated pathway results)

```
# 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
```

We can now look in more details at the three pathways. The 'pathview()' func. will take the KEGG pathway ID (printed first above) and our vector of importance and annotate the pathway with our genes.

First, we will look at hsa05310 Asthma

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

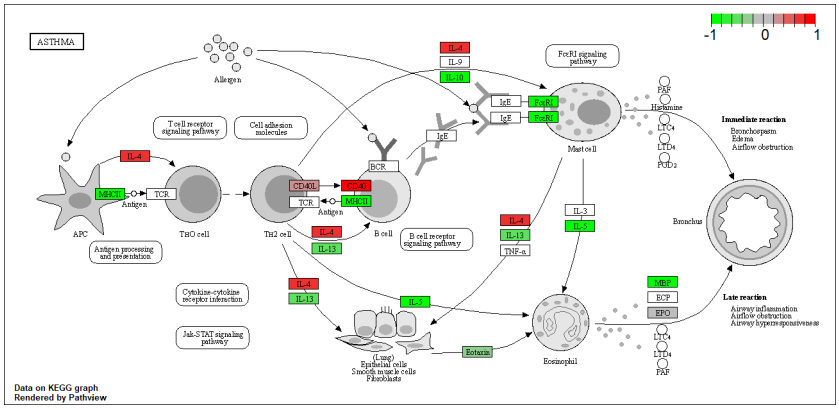


Figure 1: The Asthma pathway with our genes colored

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

Ans12:

Look at hsa05332 Graft-versus-host disease

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

Look at hsa04940 Type I diabetes mellitus

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

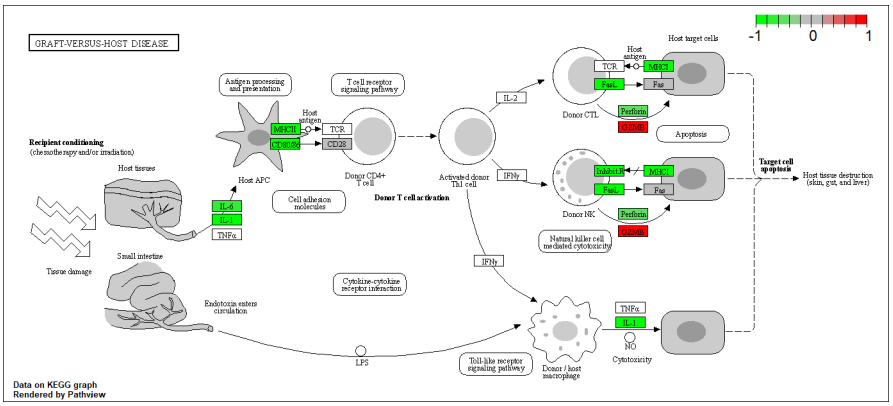


Figure 2: The Graft-versus-host disease pathway with our genes colored

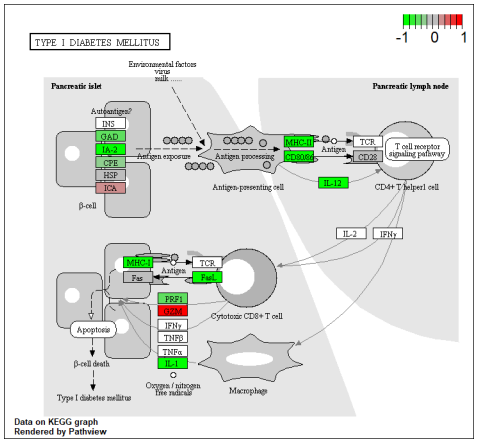


Figure 3: The Type I diabetes mellitus pathway with our genes colored