

Transcriptomics and the analysis of RNA-Seq data

Ethan Harding (PID A15468670)

11/16/2021

Import and read countData and colData

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

Let's have a look at these

```
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
```

```
##      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
## 7 SRR1039520 control N061011 GSM1275874
## 8 SRR1039521 treated N061011 GSM1275875
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
## [1] 38694
```

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

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

```
## [1] 4
```

#Toy Differential Gene Expression

First, we need to extract all the “control” columns. Then I will take the rowwise mean to get the average count values for all genes in these four experiments.

```
control.ind <- metadata$dex == "control"  
control.counts <- counts[, control.ind]  
head(control.counts)
```

```
##          SRR1039508 SRR1039512 SRR1039516 SRR1039520  
## ENSG000000000003    723      904     1170      806  
## ENSG000000000005      0       0       0       0  
## ENSG00000000419     467      616      582      417  
## ENSG00000000457     347      364      318      330  
## ENSG00000000460      96       73      118      102  
## ENSG00000000938      0        1        2        0
```

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

```
## ENSG000000000003 ENSG000000000005 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?

Rather than using “Rowsums” then diving by 4, use “Rowmeans”, that way if we were to add more data (aka more rows), the code would still work for the datasets.

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) Do the same thing for the “treated” columns.

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

```

##          SRR1039509 SRR1039513 SRR1039517 SRR1039521
## ENSG000000000003      486       445      1097      604
## ENSG000000000005       0         0         0         0
## ENSG000000000419     523       371      781      509
## ENSG000000000457     258       237      447      324
## ENSG000000000460      81        66       94       74
## ENSG000000000938      0         0         0         0

treated.mean <- rowMeans(treated.counts)
head(treated.mean)

## ENSG000000000003 ENSG000000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460
##       658.00           0.00      546.00      316.50      78.75
## ENSG000000000938
##       0.00

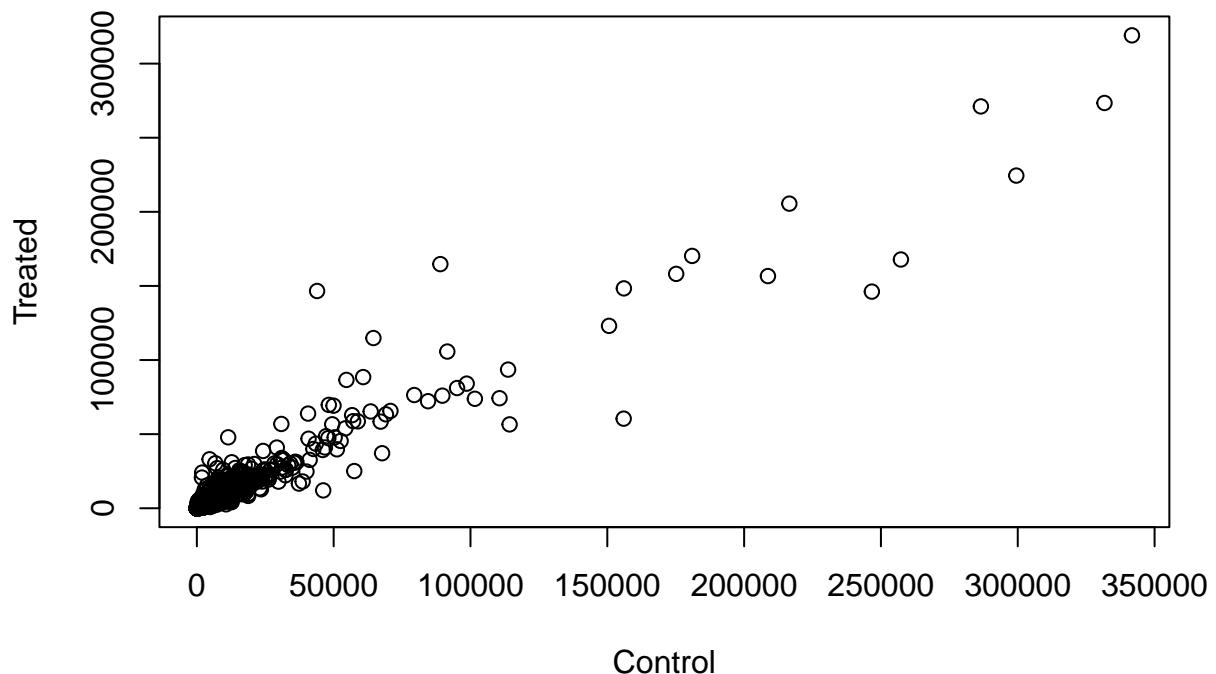
```

Combine our meancount data for bookkeeping purposes.

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

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.

```
plot(meancounts, xlab="Control", ylab="Treated")
```

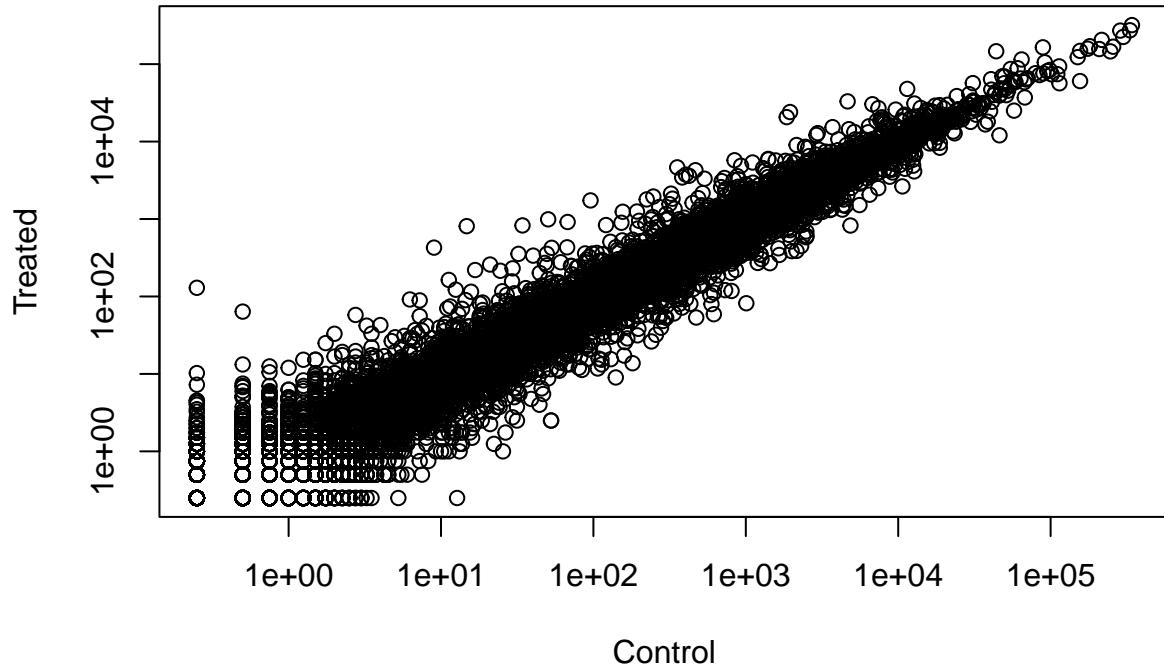


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

```
plot(meancounts, xlab="Control", ylab="Treated", 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 see 0 values for no change, + values for increases and - values for decreases. This nice property leads us to work with **log2(fold-change)** all the time in the genomics and proteomics field.

Let's add the **log2(fold-change)** to our **meancounts** dataframe.

```
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
## 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 the data to remove the genes with zero expression.

```
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?

I can use the **which()** function with the **arr.ind=TRUE** argument to get the columns and rows where the TRUE values are (i.e the zero counts in our case). Using the **unique()** function ensured we do not count any same row twice.

How many genes do we have left after removing those with zero expression?

```
nrow(mycounts)
```

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

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)
```

Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level?

```
sum(up.ind)
```

```
## [1] 250
```

Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level?

```
sum(down.ind)
```

```
## [1] 367
```

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

No, because we do not know if the fold-change levels are statistically different.

DESeq2 Analysis

Load DESeq2.

```
library(DESeq2)

## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
## 
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## 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

```

We need to first setup the input for DESeq2.

```
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): ENSG000000000003 ENSG000000000005 ... ENSG00000283120
##   ENSG00000283123
## rowData names(0):
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(4): id dex celltype geo_id

```

Now we can run DESeq 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 at the results, we use the deseq `results()` function:

```

res <- results(dds)
head(res)

## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 6 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
##             <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG000000000003  747.194195    -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005   0.000000       NA        NA        NA        NA
## ENSG00000000419   520.134160    0.2061078  0.101059  2.039475 0.0414026
## ENSG00000000457   322.664844    0.0245269  0.145145  0.168982 0.8658106
## ENSG00000000460   87.682625    -0.1471420  0.257007 -0.572521 0.5669691
## ENSG00000000938   0.319167    -1.7322890  3.493601 -0.495846 0.6200029
##           padj
##             <numeric>
## ENSG000000000003  0.163035
## ENSG000000000005   NA
## ENSG00000000419   0.176032
## ENSG00000000457   0.961694
## ENSG00000000460   0.815849
## ENSG00000000938   NA

```

Save our results

Write out whole results dataset (including genes that don't change significantly).

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

Focus in on those genes with a small p-value (i.e. show a significant change).

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

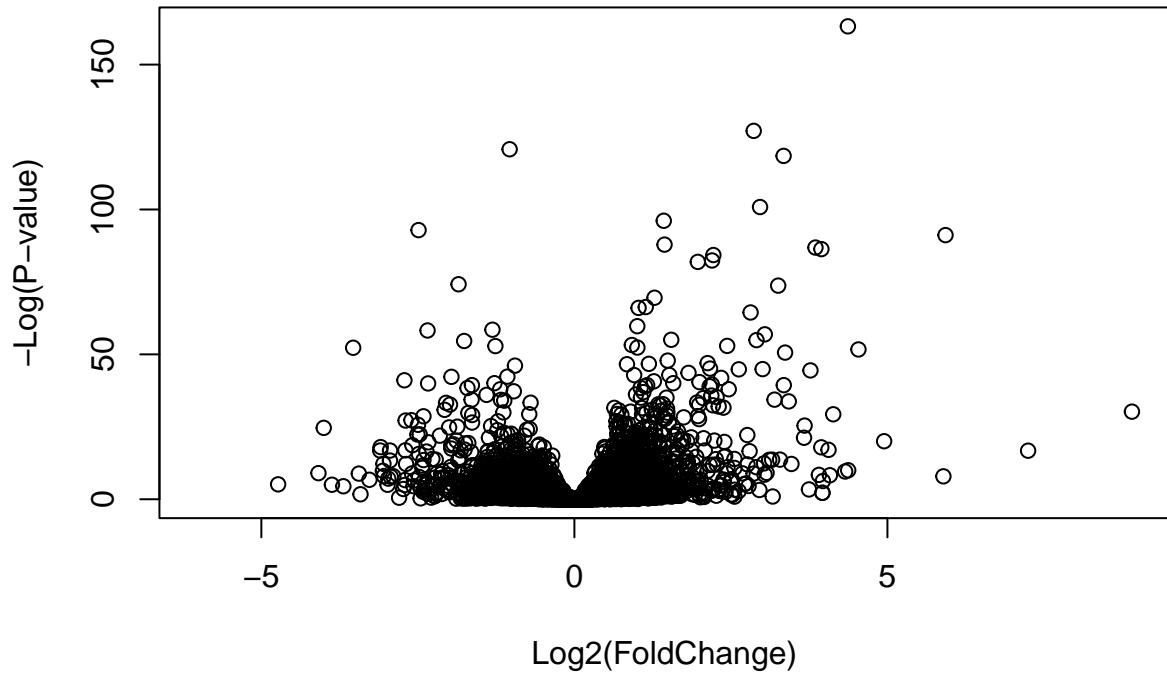
```
summary(res05)
```

```
##  
## 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
```

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

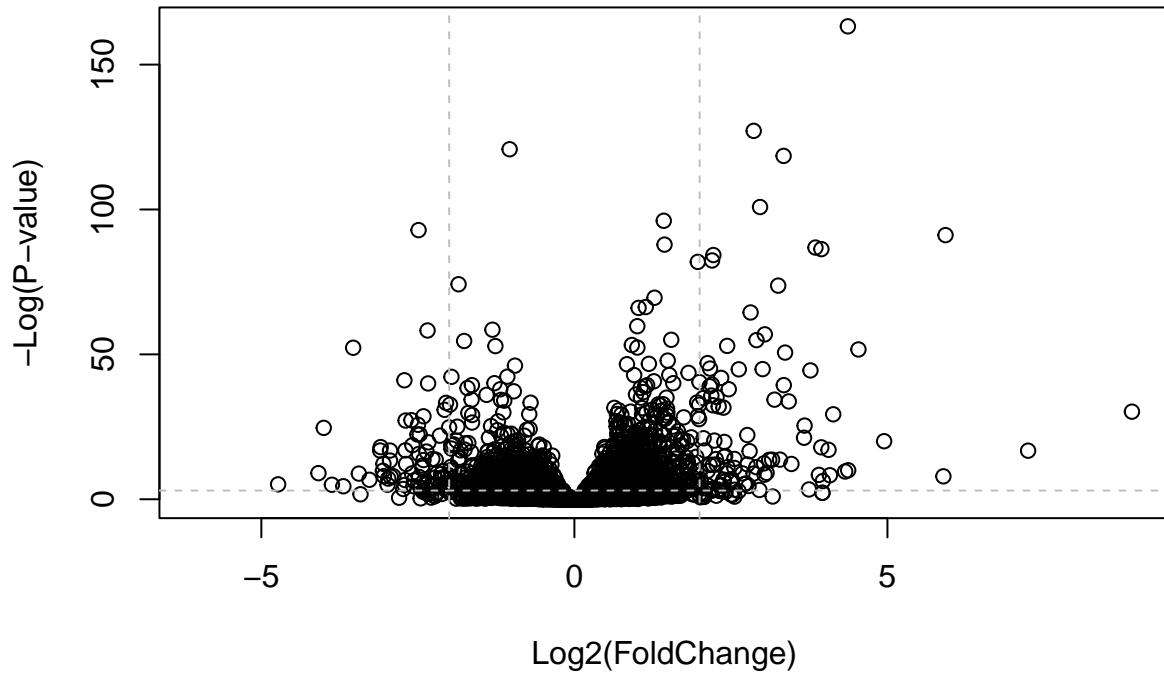


Finally, let's add some color to this plot to draw attention to the genes (i.e. points) we care about - that is those with large fold-change and low pvalues (high $-\log(p\text{value})$).

To make this more useful we can add some guidelines (with the abline() function) and color (with a custom color vector) highlighting genes that have $\text{padj} < 0.05$ and the absolute $\text{log2FoldChange} > 2$.

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

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



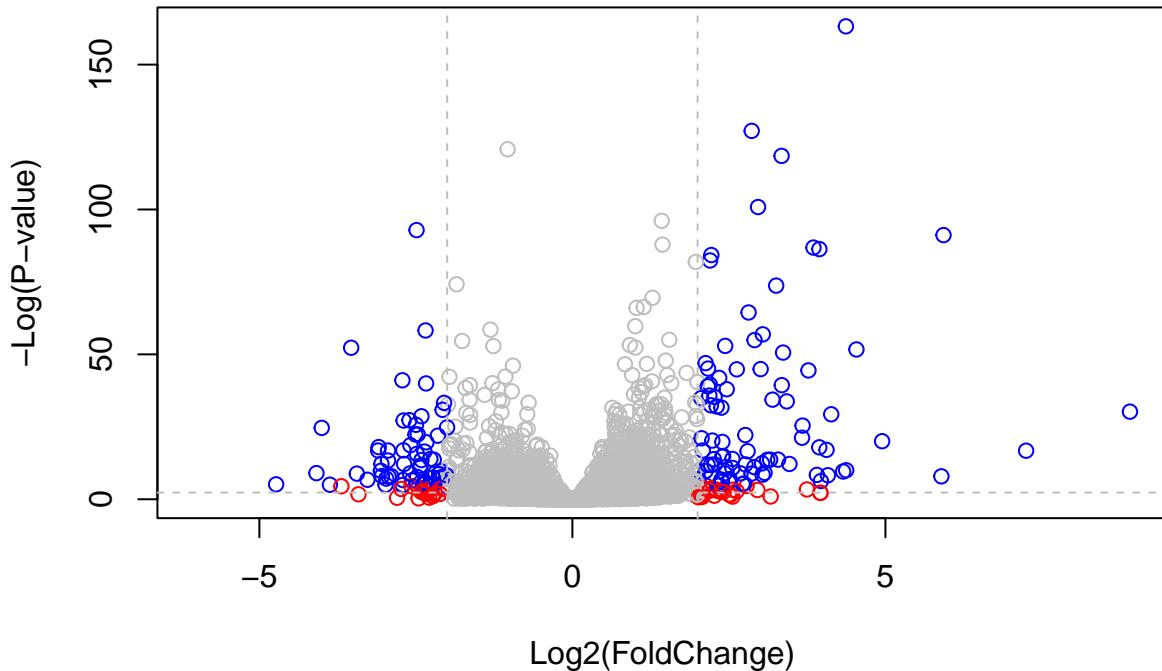
To color the points we will setup a custom color vector indicating transcripts with large fold change and significant differences between conditions:

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



```
# Add annotation data for our genes.
```

For this we need two bioconductor packages: “AnnotationDbi” and “org.Hs.eg.db”.

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

```
##
```

Let's have a look at what is in the `org.Hs.eg.db` package.

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

```
## [1] "ACCNUM"      "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 identifiers from different 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
                      multiVals="first")
```

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

Now, we can see the new column that we added.

```
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    C1orf112
## ENSG00000000938   NA        FGR
```

We need Entrez IDs for pathway analysis with KEGG.

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res\$entrez, res\$uniprot and res\$genename.

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

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

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

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

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

```

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

head(res)

## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 10 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
##           <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG000000000003 747.194195    -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005  0.000000        NA        NA        NA        NA
## ENSG000000000419 520.134160     0.2061078  0.101059  2.039475 0.0414026
## ENSG000000000457 322.664844     0.0245269  0.145145  0.168982 0.8658106
## ENSG000000000460 87.682625     -0.1471420  0.257007 -0.572521 0.5669691
## ENSG000000000938 0.319167     -1.7322890  3.493601 -0.495846 0.6200029
##           padj      symbol      entrez      uniprot
##           <numeric> <character> <character> <character>
## ENSG000000000003 0.163035      TSPAN6      7105 AOA024RCIO
## ENSG000000000005   NA        TNMD      64102 Q9H2S6
## ENSG000000000419 0.176032      DPM1      8813 060762
## ENSG000000000457 0.961694      SCYL3      57147 Q8IZE3
## ENSG000000000460 0.815849      C1orf112    55732 AOA024R922
## ENSG000000000938   NA        FGR       2268 P09769
##           genename
##           <character>
## ENSG000000000003      tetraspanin 6
## ENSG000000000005      tenomodulin
## ENSG000000000419 dolichyl-phosphate m..
## ENSG000000000457 SCY1 like pseudokina..
## ENSG000000000460 chromosome 1 open re..
## ENSG000000000938 FGR proto-oncogene, ..

```

Let's make another volcano plot with the same gene labels. For this we can use the EnhancedVolcano package.

```

library(EnhancedVolcano)

## Loading required package: ggplot2

## Loading required package: ggrepel

## Registered S3 methods overwritten by 'ggalt':
##   method             from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob ggplot2
##   grobX.absoluteGrob    ggplot2
##   grobY.absoluteGrob    ggplot2

x <- as.data.frame(res)

```

```

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

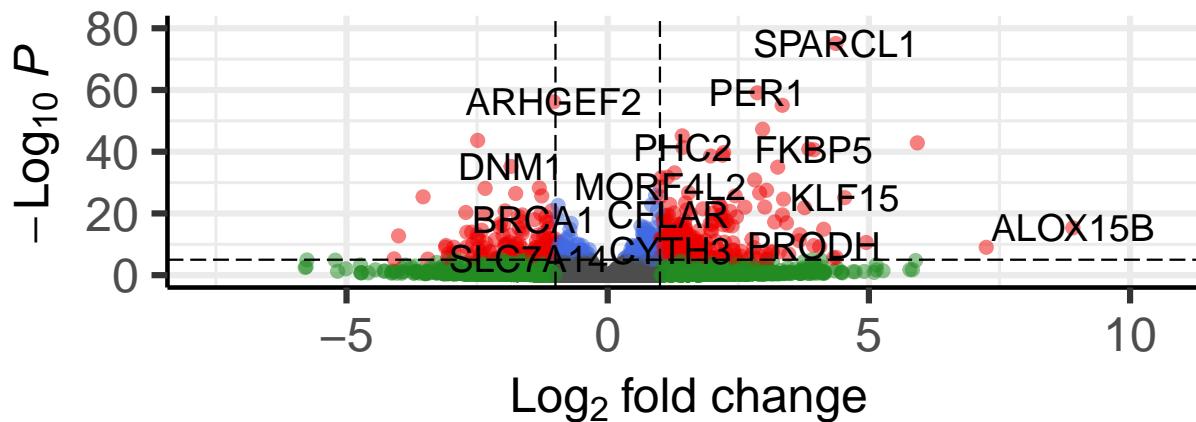
## Warning: Ignoring unknown parameters: xlim, ylim

```

Volcano plot

EnhancedVolcano

● NS ● Log₂ FC ● p-value ● p – value and log₂ FC



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)

# 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()* function requires a name vector of fold changes, where the names of the values are the Entrez gene ID's.

```

#res$entrez
foldchanges <- res$log2FoldChange
names(foldchanges) <- res$entrez

head(foldchanges)

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

```

```

# get the results
keggres <- gage(foldchanges, gsets=kegg.sets.hs)

attributes(keggres)

```

```

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

```

This separates out results by “greater” and “less” (i.e. those that are up regulated and those that are down regulated).

```

# 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

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

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

## Info: Working in directory /Users/ethanharding/Desktop/BIMM 143/bimm143_github/class15

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

```

