

# Lab 13

Adam Bisharat

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

Loading required package: S4Vectors

Loading required package: stats4

Loading required package: BiocGenerics

Attaching package: 'BiocGenerics'

The following objects are masked from 'package:stats':

IQR, mad, sd, var, xtabs

The following objects are masked from 'package:base':

anyDuplicated, aperm, 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, saveRDS, setdiff,  
table, tapply, union, unique, unsplit, which.max, which.min

Attaching package: 'S4Vectors'

```
The following object is masked from 'package:utils':
```

```
  findMatches
```

```
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
```

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

```
[1] 38694
```

```
head(counts)
```

|                  | SRR1039508 | SRR1039509 | SRR1039512 | SRR1039513 | SRR1039516 |
|------------------|------------|------------|------------|------------|------------|
| ENSG000000000003 | 723        | 486        | 904        | 445        | 1170       |
| ENSG000000000005 | 0          | 0          | 0          | 0          | 0          |
| ENSG00000000419  | 467        | 523        | 616        | 371        | 582        |
| ENSG00000000457  | 347        | 258        | 364        | 237        | 318        |
| ENSG00000000460  | 96         | 81         | 73         | 66         | 118        |
| ENSG00000000938  | 0          | 0          | 1          | 0          | 2          |
|                  | SRR1039517 | SRR1039520 | SRR1039521 |            |            |
| ENSG000000000003 | 1097       | 806        | 604        |            |            |
| ENSG000000000005 | 0          | 0          | 0          |            |            |
| ENSG00000000419  | 781        | 417        | 509        |            |            |
| ENSG00000000457  | 447        | 330        | 324        |            |            |
| ENSG00000000460  | 94         | 102        | 74         |            |            |
| ENSG00000000938  | 0          | 0          | 0          |            |            |

```
View(metadata)
```

Q1. How many genes are in this dataset?

There are 38694 genes

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

```
[1] 4
```

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

```
control treated
      4       4
```

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

There are 4 control cell lines

### Toy differential expression analysis

Calculate the mean per gene count values for all “control” samples (i.e columns in `counts`) and do the same for “treated” and then compare them.

1. Find all “control values/columns in `counts`

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

```
FALSE  TRUE
      4       4
```

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

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

2. Find the mean per gene across all control columns.

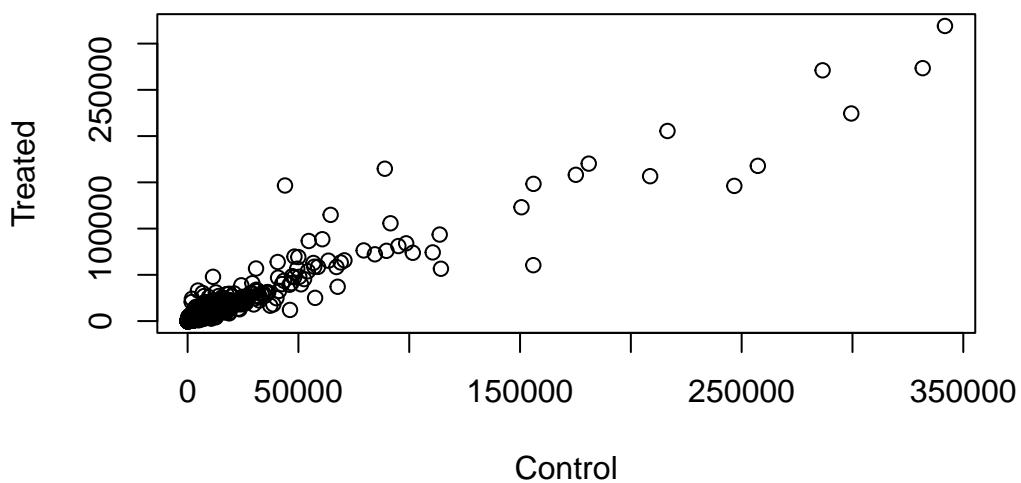
```
control.mean <- apply(control.counts, 1, mean)
```

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 inds <- metadata$dex == "treated"  
treated.counts <- counts[,treated inds]  
treated.mean <- apply(treated.counts, 1, mean)
```

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.

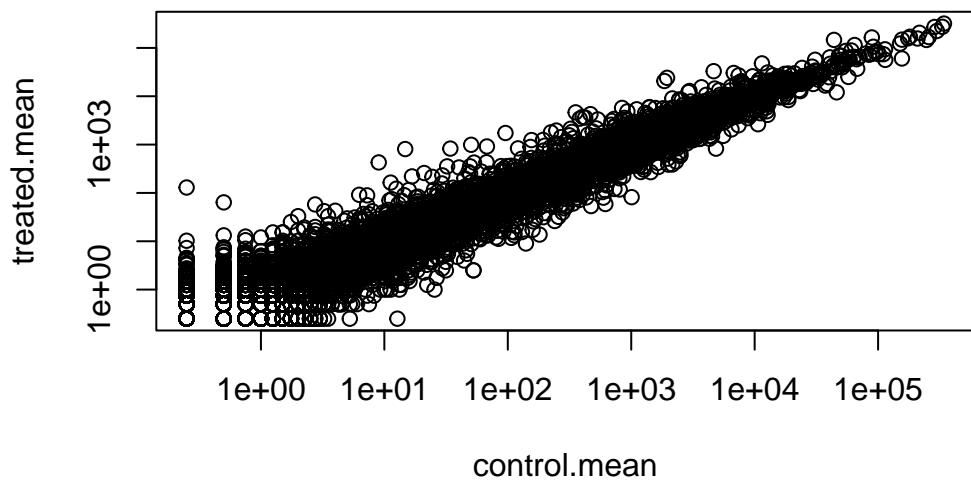
```
meancounts <- data.frame(control.mean, treated.mean)  
plot(meancounts[,1], meancounts[,2], xlab="Control", ylab="Treated")
```



```
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
```



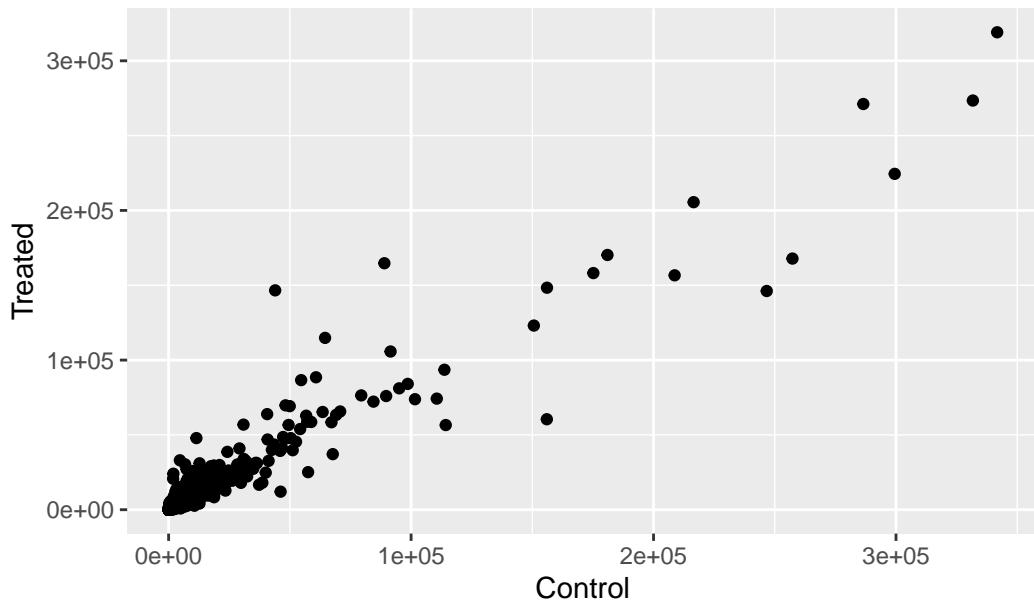
Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom\_?() function would you use for this plot?

point

```
library(ggplot2)

ggplot(meancounts, aes(x = control.mean, y = treated.mean)) +
  geom_point() +
  xlab("Control") +
  ylab("Treated") +
  ggtitle("Control vs Treated Mean Counts")
```

## Control vs Treated Mean Counts



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

```
plot(meancounts, log='xy')
```

We most frequently use log2 transformations for this type of data

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

```
[1] 0
```

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

```
[1] 1
```

```
log2(30/10)
```

```
[1] 1.584963
```

```
log2(40/10)
```

```
[1] 2
```

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

```
[1] -1
```

These log2 values make the interpretation of “fold-change” a little easier and a rule-of-thumb in the field is a log2 fold-change of +2 or -2 is where we start to pay attention

```
log2(40/10)
```

```
[1] 2
```

Lets calculate the log2 (fold-change) and add it to our `meancounts` data.frame

```
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        |

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?

```
to.rm <- rowSums((meancounts[,1:2]==0) > 0)
mycounts <- meancounts[!to.rm,]
```

Q. how many genes do I have left after this zero count filtering

```
nrow(mycounts)
```

```
[1] 21817
```

Q8. How many genes are “up” regulated upon drug treatment with a threshold of +2 log2-fold-change?

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

[1] 250

250 up-regulated genes

Q9. How many genes are “down” regulated upon drug treatment with a threshold of -2 log2-fold-change?

```
down.ind <- mycounts$log2fc < (-2)
count(down.ind)
```

[1] 367

367 down regulated genes

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

No, there is no statistics to analyze if these results are statistically significant.

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

The first function that we will use will setup the data in the way (format) DESeq wants it.

```
dds <- DESeqDataSetFromMatrix(countData=counts,
                               colData=metadata,
                               design=~dex)
```

converting counts to integer mode

Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in  
design formula are characters, converting to factors

```
dds
```

```
class: DESeqDataSet
dim: 38694 8
metadata(1): version
assays(1): counts
rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
  ENSG00000283123
rowData names(0):
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(3): dex celltype geo_id
```

The function in the package is called `DESeq()`, and we can run it on our `dds` object.

```
DESeq(dds)
```

estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

```
class: DESeqDataSet
dim: 38694 8
metadata(1): version
assays(4): counts mu H cooks
rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
  ENSG00000283123
rowData names(22): baseMean baseVar ... deviance maxCooks
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(4): dex celltype geo_id sizeFactor
```

```
dds <- estimateSizeFactors(dds)
sizeFactors(dds)
```

```
SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520  
1.0193796 0.9005653 1.1784239 0.6709854 1.1731984 1.3929361 0.9180030  
SRR1039521  
0.9504390
```

```
dds <- estimateDispersions(dds)
```

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

```
#dispersions(dds)
```

```
dds <- nbinomWaldTest(dds)
```

```
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

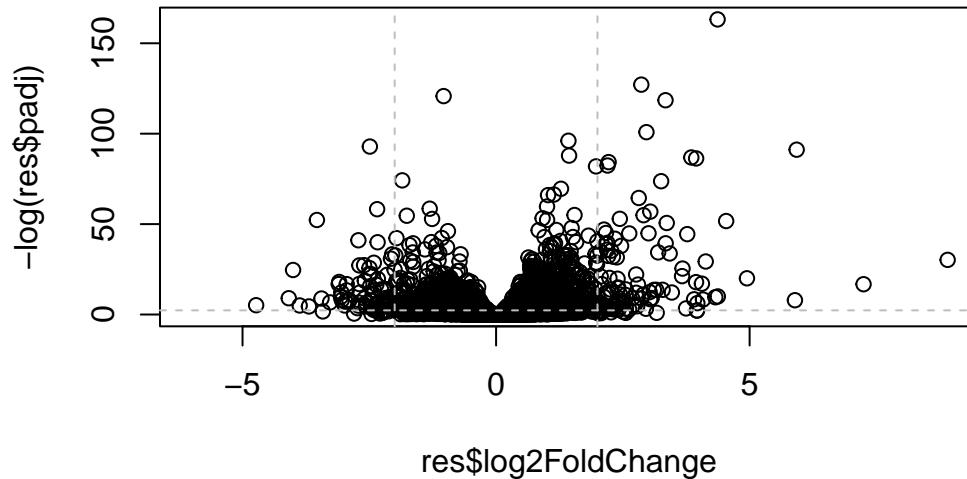
final dispersion estimates

fitting model and testing

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

Make a common overall results figure from this analysis. This is designed to keep our inner biologist and inner stats nerd happy - It plots fold -change on x-axis vs p-value on y axis

```
plot( res$log2FoldChange, -log(res$padj) )
abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)
```

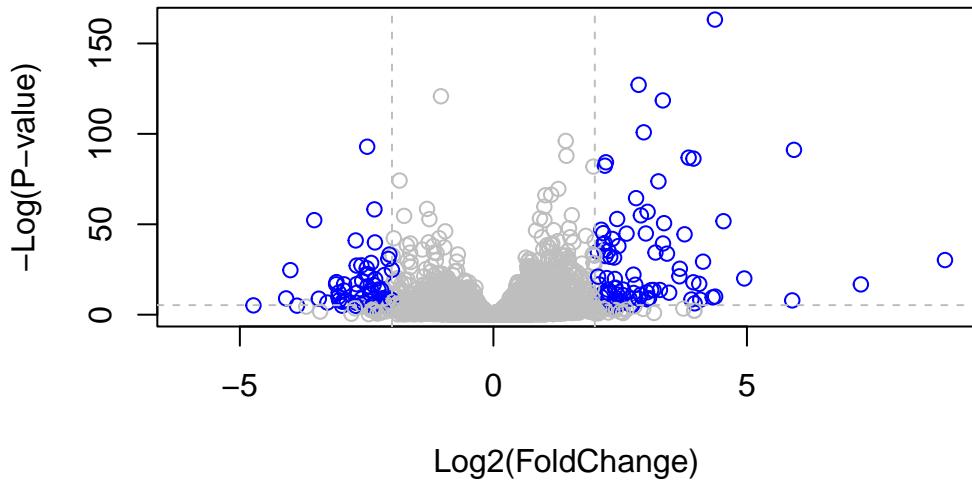


Add some color:

```
mycols <- rep("gray", nrow(res))
mycols[res$log2FoldChange > 2 ] <- "red"
mycols[res$log2FoldChange < -2 ] <- "red"
mycols[res$padj >.005] <- "grey"

inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"

plot( res$log2FoldChange, -log(res$padj),
      col=mycols, ylab="-Log(P-value)", xlab="Log2(FoldChange)" )
abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.005), col="gray", lty=2)
```



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

```
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
```

I need to translate our gene identifiers “ENSG000...” into the gene names that the rest of the world can understand.

To this “annotation” I will use the “**AnnotationDbi**” package. I can install this with `BiocManager::install()`

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

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

```
[1] "ACCCNUM"          "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"
```

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

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

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 7 columns
  baseMean log2FoldChange      lfcSE      stat      pvalue
  <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.3507030  0.168246 -2.084470 0.0371175
ENSG000000000005  0.000000    NA        NA        NA        NA
ENSG00000000419   520.134160  0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457   322.664844  0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460   87.682625 -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167 -1.7322890  3.493601 -0.495846 0.6200029
  padj      symbol
  <numeric> <character>
ENSG000000000003  0.163035    TSPAN6
ENSG000000000005    NA        TNMD
```

```

ENSG000000000419 0.176032      DPM1
ENSG000000000457 0.961694      SCYL3
ENSG000000000460 0.815849      FIRRM
ENSG000000000938 NA            FGR

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

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

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

'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>
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      genename      entrez
  <numeric> <character> <character> <character>
ENSG000000000003 0.163035    TSPAN6       tetraspanin 6      7105
ENSG000000000005 NA          TNMD        tenomodulin   64102
ENSG000000000419 0.176032    DPM1 dolichyl-phosphate m.. 8813
ENSG000000000457 0.961694    SCYL3 SCY1 like pseudokina.. 57147
ENSG000000000460 0.815849    FIRRM FIGNL1 interacting r.. 55732
ENSG000000000938 NA          FGR FGR proto-oncogene, ... 2268

```

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

Now that we have our results with added annotation we can do some pathway mapping

Let's use the **gage** package to look for KEGG pathways in our results (genes of interest). I will also use the **pathview** package to draw little pathway figures.

```
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"
```

What **gage** wants as input is not my big table/data.frame of results. It just wants a “vector of importance.” For RNASeq data like we have this is our log2FC values...

```
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
```

```
keggres = gage(foldchanges, gsets = kegg.sets.hs)
```

```
attributes(keggres)
```

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

```
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 |

Let's use the pathview package to look at one of these highlighted KEGG pathways with our genes highlighted.

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

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

```
Info: Working in directory /Users/adambish/Lab13
```

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

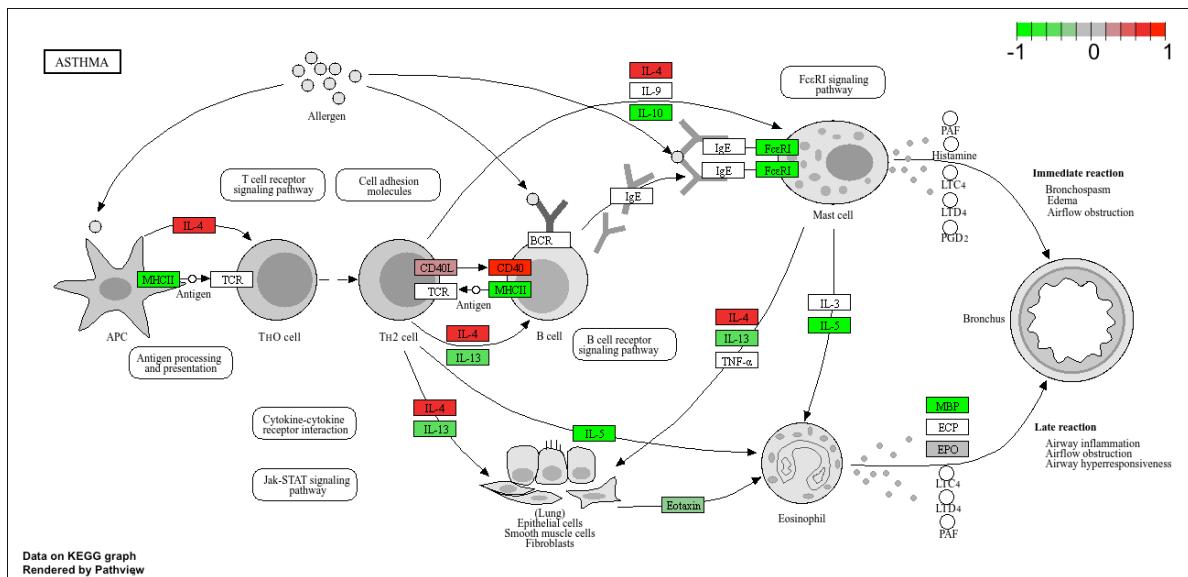
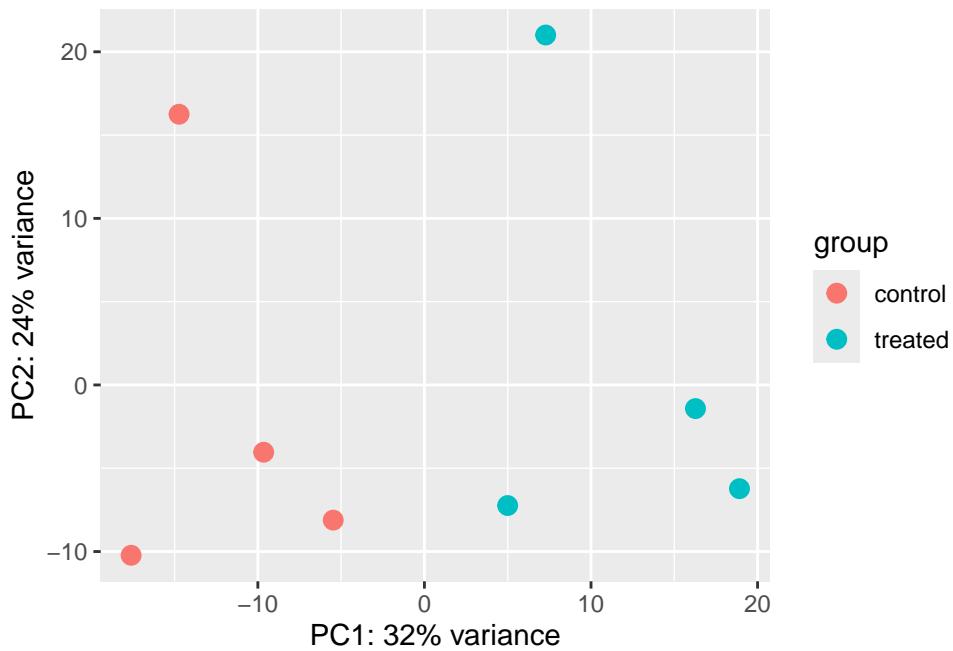


Figure 1: Asthma Pathway with my DEGs

```
vsd <- vst(dds, blind = FALSE)
plotPCA(vsd, intgroup = c("dex"))
```

using ntop=500 top features by variance



```
pcaData <- plotPCA(vsd, intgroup=c("dex"), returnData=TRUE)
```

using ntop=500 top features by variance

```
head(pcaData)
```

|            | PC1        | PC2        | group   | dex     | name       |
|------------|------------|------------|---------|---------|------------|
| SRR1039508 | -17.607922 | -10.225252 | control | control | SRR1039508 |
| SRR1039509 | 4.996738   | -7.238117  | treated | treated | SRR1039509 |
| SRR1039512 | -5.474456  | -8.113993  | control | control | SRR1039512 |
| SRR1039513 | 18.912974  | -6.226041  | treated | treated | SRR1039513 |
| SRR1039516 | -14.729173 | 16.252000  | control | control | SRR1039516 |
| SRR1039517 | 7.279863   | 21.008034  | treated | treated | SRR1039517 |

```
percentVar <- round(100 * attr(pcaData, "percentVar"))
```

```
ggplot(pcaData) +  
  aes(x = PC1, y = PC2, color = dex) +  
  geom_point(size = 3) +  
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +  
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +  
  coord_fixed() +  
  theme_bw()
```

