

Lab 13 Transcriptomics

Daniel Gurholt (PID: A16767491)

Today we will analyze some RNASeq data from Hines et al. on the effects of dexamethasone (dex), a synthetic glucocorticoid steroid with anti-inflammatory effects, in four different ASM cell lines treated with dexamethasone glucocorticoid.

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

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG000000000419	467	523	616	371	582
ENSG000000000457	347	258	364	237	318
ENSG000000000460	96	81	73	66	118
ENSG000000000938	0	0	1	0	2
	SRR1039517	SRR1039520	SRR1039521		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG000000000419	781	417	509		
ENSG000000000457	447	330	324		
ENSG000000000460	94	102	74		
ENSG000000000938	0	0	0		

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes in this data set

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

```
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
```

```
ctrl<- metadata$dex=="control"
sum(ctrl)
```

```
[1] 4
```

We have 4 control/untreated cell lines in the metadata dataset

```
##Toy differential analysis
```

Calculate the mean per gene count values for all control samples and all treated samples and then compare them

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

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

	SRR1039508	SRR1039512	SRR1039516	SRR1039520
ENSG00000000003	723	904	1170	806
ENSG00000000005	0	0	0	0
ENSG00000000419	467	616	582	417
ENSG00000000457	347	364	318	330
ENSG00000000460	96	73	118	102
ENSG00000000938	0	1	2	0

2. Find the mean per gene across all control columns

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

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

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

To make the code more robust we can add the apply function to utilize the code in all columns no matter how many columns there are or is added which can make it useful for different datasets without changing the code.

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)

3. Find all “treated” values/columns in counts

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

4. Find all “treated” values/columns in counts

```
treated.mean<- apply(treated.counts, 1, mean)
head(treated.mean)
```

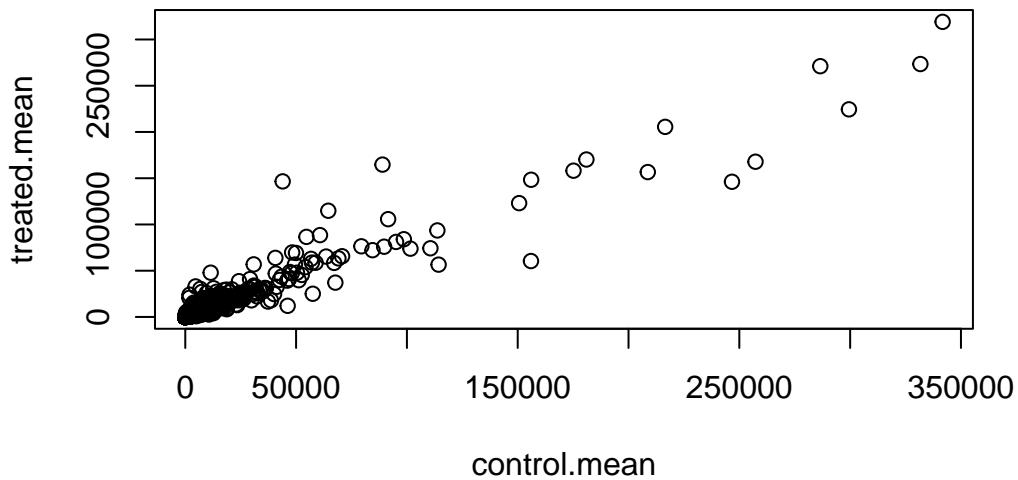
ENSG000000000003	ENSG000000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
658.00	0.00	546.00	316.50	78.75
ENSG000000000938				
0.00				

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

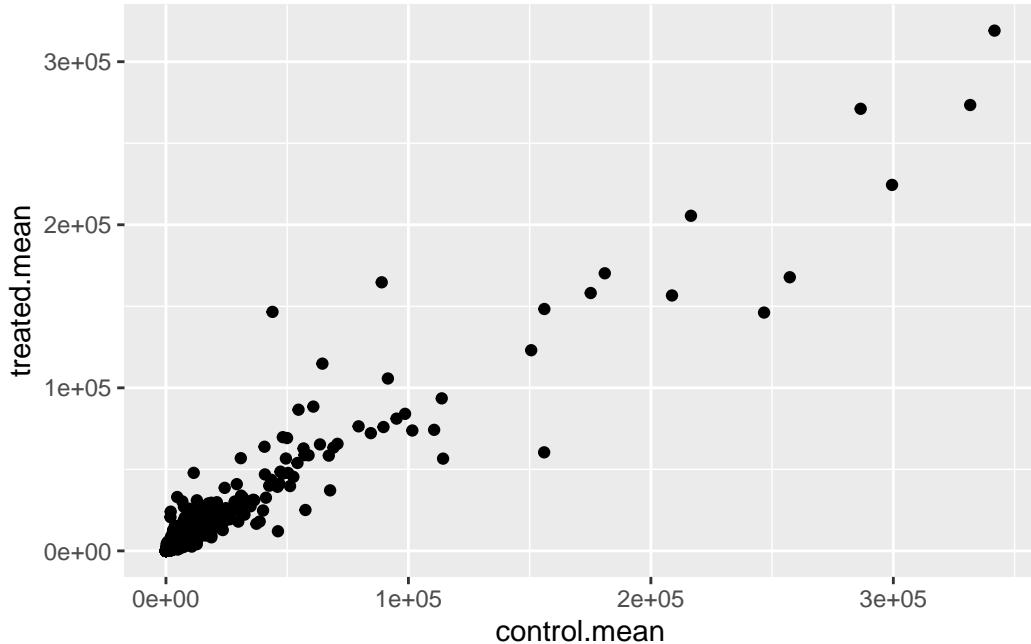


Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

You would use geom_point for this ggplot

```
library(ggplot2)

ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  geom_point()
```



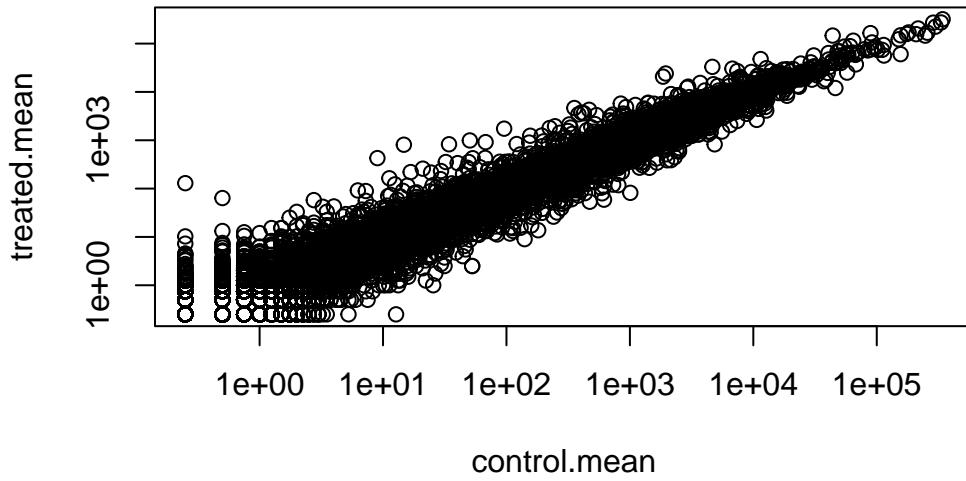
Q6. Try plotting both axes on a log scale. What is the argument to `plot()` that allows you to do this?

The `Log` argument to `plot()` allows you to do this

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

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted  
from logarithmic plot
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted  
from logarithmic plot
```



We most frequently use log2 transformations for this type of data

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

```
[1] 0
```

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

```
[1] 1
```

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

```
[1] -1
```

These log2 values make the interpretation of “foldchange” 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(40/20)
```

```
[1] 1
```

Let's 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

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

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

	control.mean	treated.mean	log2fc
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000971	5219.00	6687.50	0.35769358
ENSG000000001036	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 purpose of the arr.ind argument in the which() function is to arrange all rows/samples in which the two columns of meancounts data show up as TRUE if there are any zero values and FALSE if there are actual nonzero numerical values so we can remove the values that do not change. The unique() function allows us to ensure that if both columns appear as zero they are not duplicated as two TRUE values so they can be removed with further code because they do not need to be analyzed.

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

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

```
[1] 250
```

There are 250 up regulated genes at the 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?

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

```
[1] 367
```

There are 367 down regulated genes at the less than than -2 fc level

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

I cannot quite trust these results yet because even though we might find a lot of samples which are down regulated and up regulated, they might not be statistically significant in their differences to the control genes so we must use DESeq analysis to confirm which in these up and down regulated group is significant.

Is this difference in the mean counts significant??? Let's do this analysis the right way with stats and use DESeq analysis.

```
##DESeq Analysis
```

```
#/ message: false  
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, 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

Attaching package: 'IRanges'

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

windows

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

```
Loading required package: MatrixGenerics

Loading required package: matrixStats

Warning: package 'matrixStats' was built under R version 4.4.2

Attaching package: 'MatrixGenerics'

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

colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
colCounts, colCummmaxs, colCummins, colCumprods, colCumsums,
colDiffss, colIQRDiffss, colIQRs, colLogSumExps, colMadDiffss,
colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
colProds, colQuantiles, colRanges, colRanks, colSdDiffss, colSds,
colSums2, colTabulates, colVarDiffss, colVars, colWeightedMads,
colWeightedMeans, colWeightedMedians, colWeightedSds,
colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
rowCollapse, rowCounts, rowCummmaxs, rowCummins, rowCumprods,
rowCumsums, rowDiffss, rowIQRDiffss, rowIQRs, rowLogSumExps,
rowMadDiffss, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
rowSdDiffss, rowSds, rowSums2, rowTabulates, rowVarDiffss, 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
```

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
```

The function in the package is called DESeq and we can run it on our dds object above

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

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

```

log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
      baseMean log2FoldChange      lfcSE      stat     pvalue
      <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.3507030  0.168246 -2.084470 0.0371175
ENSG000000000005  0.000000      NA        NA        NA        NA
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

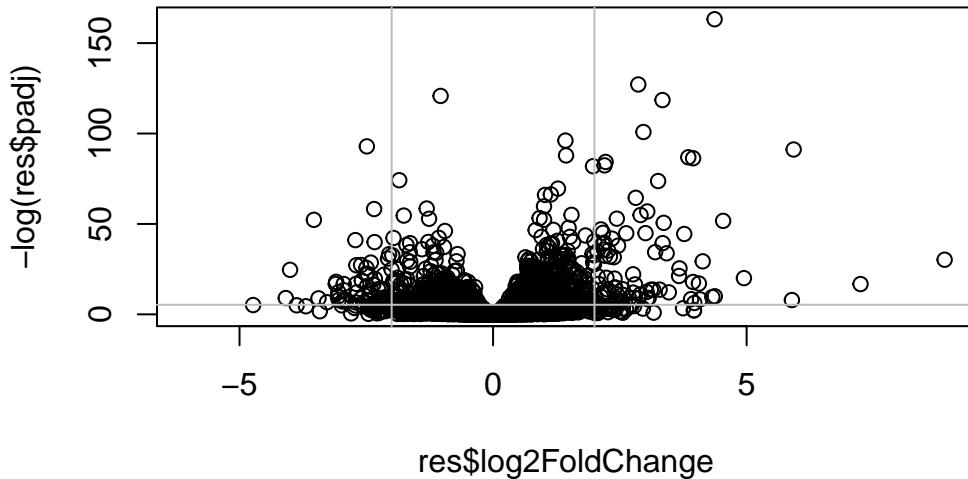
```

Make a common overall results figure from this analysis. This is designed to keep our inner biologists and our inner stats nerd happy. It plots fold-chnage vs P-value

```

plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2,2), col="gray")
abline(h=-log(0.005), col="gray")

```



Add some color to this plot:

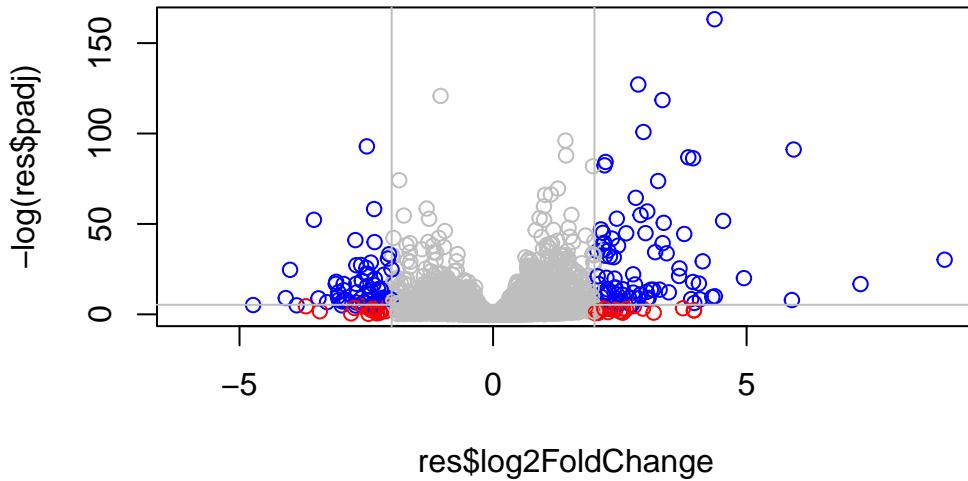
```

mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

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

plot(res$log2FoldChange, -log(res$padj), col=mycols)
abline(v=c(-2,2), col="gray")
abline(h=-log(0.005), col="gray")

```



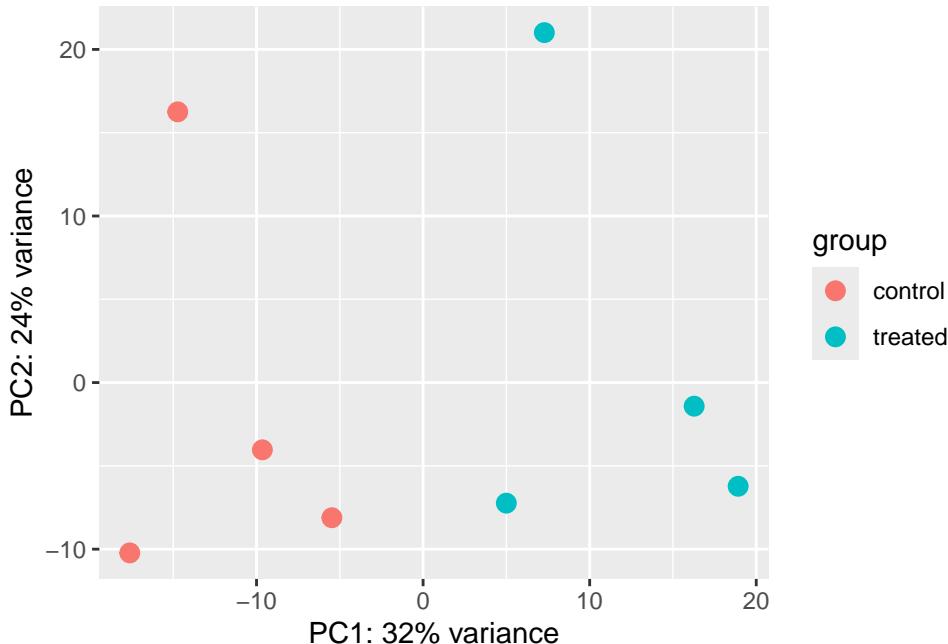
I want to save my results to date out to disc

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

```
##Principal Component Analysis (PCA)
```

```
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

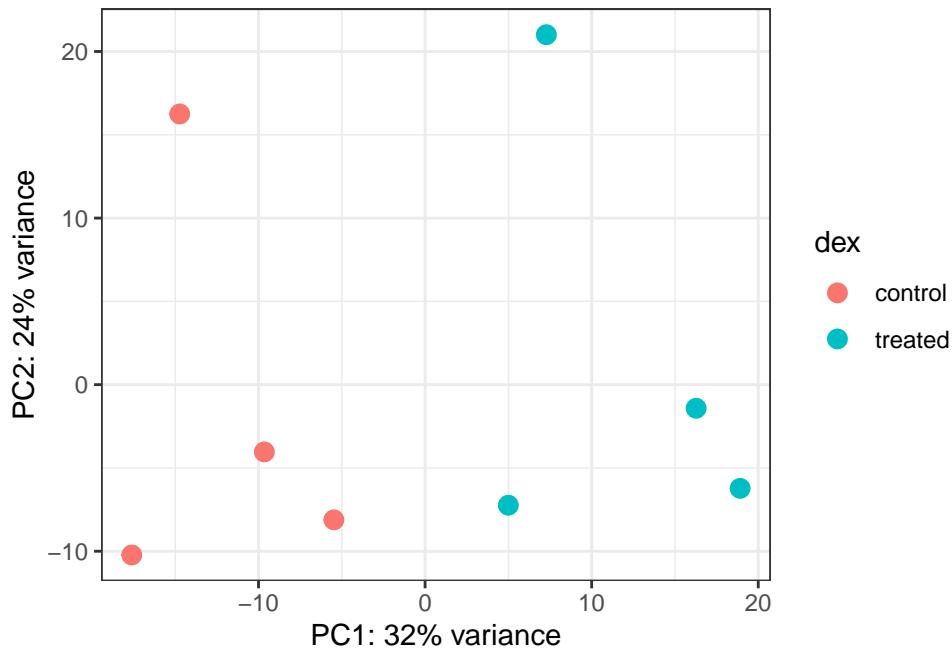
```
# Calculate percent variance per PC for the plot axis labels
percentVar <- round(100 * attr(pcaData, "percentVar"))
```

```
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()

```



I need to translate our gene identifiers “ENSG00000003” into gene names that the rest of the world can understand.

To do this annotation, I will use the “AnnotationDbi” package via BiocManager

```

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

```

```

columns(org.Hs.eg.db)

```

```

[1] "ACNUM"          "ALIAS"           "ENSEMBL"         "ENSEMLPROT"      "ENSEMLTRANS"
[6] "ENTREZID"       "ENZYME"          "EVIDENCE"        "EVIDENCEALL"    "GENENAME"
[11] "GENETYPE"       "GO"               "GOALL"           "IPI"             "MAP"
[16] "OMIM"            "ONTOLOGY"        "ONTOLOGYALL"    "PATH"            "PFAM"
[21] "PMID"           "PROSITE"          "REFSEQ"          "SYMBOL"          "UCSCKG"
[26] "UNIPROT"

```

I will use the `mapIds()` function to map my identifiers to those from different databases. I will go between “ENSEMBL” and “SYMBOL” (and then after “GENENAME”)

```
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
ENSG000000000003	747.194195	<numeric>	<numeric>	<numeric>	<numeric>
ENSG000000000005	0.000000	-0.3507030	0.168246	-2.084470	0.0371175
ENSG000000000419	520.134160	NA	NA	NA	NA
ENSG000000000457	322.664844	0.2061078	0.101059	2.039475	0.0414026
ENSG000000000460	87.682625	0.0245269	0.145145	0.168982	0.8658106
ENSG000000000938	87.682625	-0.1471420	0.257007	-0.572521	0.5669691
	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			

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$genename<- mapIds(org.Hs.eg.db,
                      keys= rownames(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 8 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
  <numeric> <character> <character>
ENSG000000000003 0.163035    TSPAN6      tetraspanin 6
ENSG000000000005   NA        TNMD      tenomodulin
ENSG000000000419 0.176032    DPM1 dolichyl-phosphate m..
ENSG000000000457 0.961694    SCYL3 SCY1 like pseudokina..
ENSG000000000460 0.815849    FIRRM FIGNL1 interacting r..
ENSG000000000938   NA        FGR FGR proto-oncogene, ..
```

```
res$entrezid<- 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	entrezid	
	<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

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

'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	genename	entrezid	
	<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
	uniprot				
	<character>				

```

ENSG000000000003 AOA087WYV6
ENSG000000000005 Q9H2S6
ENSG000000000419 H0Y368
ENSG000000000457 X6RHX1
ENSG000000000460 A6NFP1
ENSG000000000938 B7Z6W7

```

```

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      genename      entrezid
      <numeric> <character> <character> <character>
ENSG00000152583 1.32441e-71    SPARCL1      SPARC like 1    8404
ENSG00000179094 6.13966e-56    PER1       period circadian reg.. 5187
ENSG00000116584 3.49776e-53    ARHGEF2    Rho/Rac guanine nucl.. 9181
ENSG00000189221 3.46227e-52    MAOA      monoamine oxidase A 4128
ENSG00000120129 1.59454e-44    DUSP1      dual specificity pho.. 1843
ENSG00000148175 1.83034e-42    STOM      stomatin          2040
      uniprot
      <character>
ENSG00000152583    B4E2Z0
ENSG00000179094    A2I2P6
ENSG00000116584    A0A8Q3SIN5
ENSG00000189221    B4DF46
ENSG00000120129    B4DRR4
ENSG00000148175    F8VSL7

```

Save our annotated results object

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

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

Let's use the `gage` package to look at KEGG pathways in our results (genes of interest)

```
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$log2FoldChange  
names(foldchanges) = res$entrezid  
head(foldchanges)
```

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

Now let's run the gage pathway analysis

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

what is in this **keggres** object?

```
attributes(keggres)
```

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

```
# 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

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 C:/Users/danie/Downloads/Daniel Gurholt BIMM143/Lab 13 Transcripts

Info: Writing image file hsa05310.pathview.png

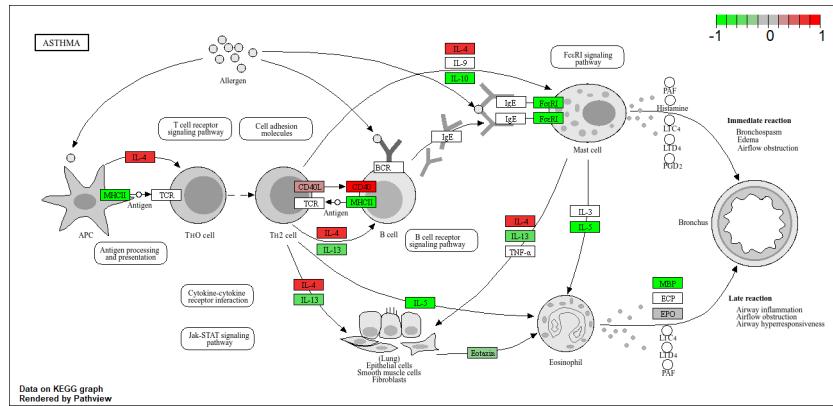


Figure 1: Asthma pathway with my DEGs

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

```
head(keggres$greater, 2)
```

		p.geomean	stat.mean	p.val
		q.val	set.size	exp1
hsa00500	Starch and sucrose metabolism	0.003306262	2.772644	0.003306262
hsa00330	Arginine and proline metabolism	0.015241247	2.194146	0.015241247
hsa00500	Starch and sucrose metabolism	0.7042337	52	0.003306262
hsa00330	Arginine and proline metabolism	0.7774866	53	0.015241247

```
pathview(gene.data=foldchanges, pathway.id="hsa00500")
```

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

Info: Working in directory C:/Users/danie/Downloads/Dani

Info: Working in directory C:/Users/danie/Downloads/Daniel Gurholt BIMM143/Lab 13 Transcripts

Info: Writing image file hsa00500.pathview.png

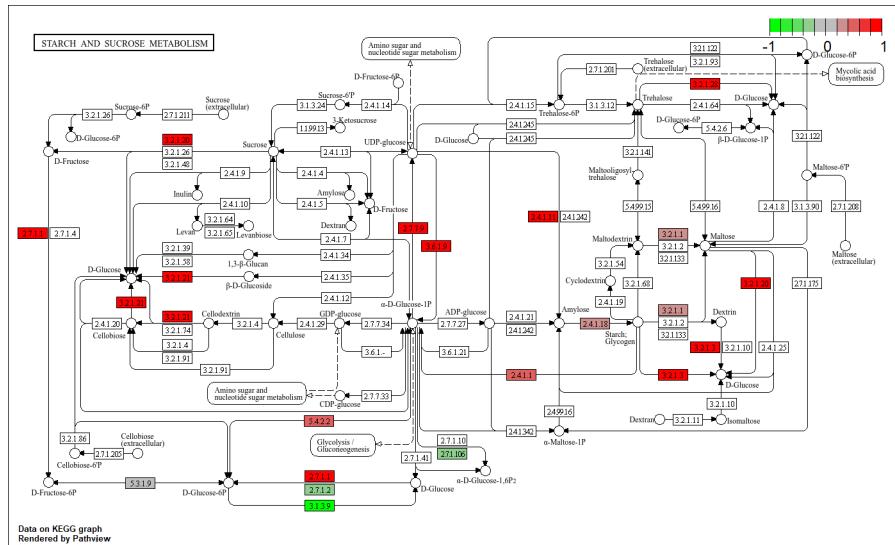


Figure 2: Down regulated pathway #1

```
pathview(gene.data=foldchanges, pathway.id="hsa00330")
```

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

```
Info: Working in directory C:/Users/danie/Downloads/Daniel Gurholt BIMM143/Lab 13 Transcript
```

```
Info: Writing image file hsa00330.pathview.png
```

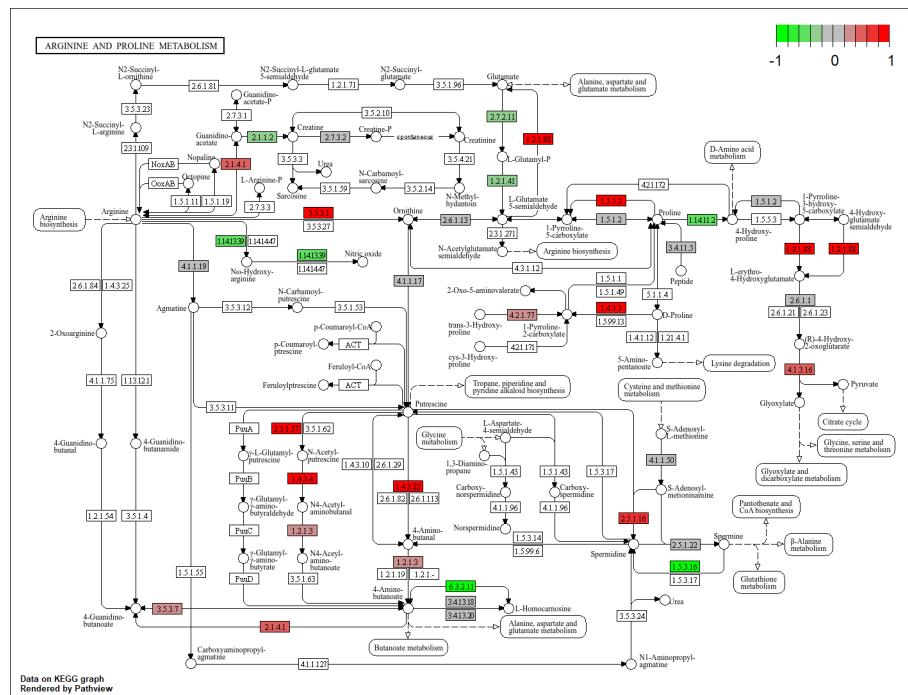


Figure 3: Down Regulated pathway #2