

Lab 15: Transcriptomics and the analysis of RNA-Seq data

forrest wang

```
##CLASS 15 #Overview Context: smooth muscle cells in airway treated with dexamethasone  
(glucocorticoid steroid, anti-inflammatory) Focus Gene: CRISPLD2
```

In class, we will work with published RNA-seq experiments where airway smooth muscles (ASMs) were treated with dexamethasone

#1. Bioconductor & DESeq2 Setups Check Installation

```
library(BiocManager)
```

```
Bioconductor version '3.15' is out-of-date; the current release version '3.16'  
is available with R version '4.2'; see https://bioconductor.org/install
```

```
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, 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,
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, rowCummaxs, 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
```

```
#2. Import countData & colData Will will used good ole' 'read.csv()' to read the two things we need for analysis: - count data - ccol data (metadata)
```

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

```
Q1.How many genes do I have? A1: 38694
```

```
nrow(counts)
```

```
[1] 38694
```

Let's have a look at the metadata Q2: How many control cell lines do we have? A2: 4 controls

```
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

And the 'counts' data

```
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		

First we should check the correspondence of the metadata & count data

```
metadata$id
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"  
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
colnames(counts)
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"  
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

To check that these are all in the same order, we can use ‘==’ test of equality.

```
metadata$id == colnames(counts)
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
all(c(T,T,T,T,T,T))
```

```
[1] TRUE
```

#3. Toy differential gene expression

##Analysis via comparison of CONTROL vs TREATED The “treated” have the dex drug and the “control” do not. 1st, I need to be able to extract just the “control” columnnes in the ‘counts’ data set

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

```
[1] "SRR1039508" "SRR1039512" "SRR1039516" "SRR1039520"
```

Now I can use this to access just the “control” column of my ‘counts’ data...

```
control.counts <- counts[,control$id]  
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

Find the mean count value for each transcript/gene by binding the ‘rowMeans()’

```
control.mean <- rowMeans(control.counts)
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? A3: Now we find a mean value for all the “treated” columns in the same way (simplified)

```
treated.id <- metadata[metadata$dex == "treated", "id"]
treated.mean <- rowMeans(counts[, treated.id])
```

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)
A4: “treated.id <- metadata[metadata\$dex == “treated”, “id”]” “treated.mean <- rowMeans(counts[,treated.id])”

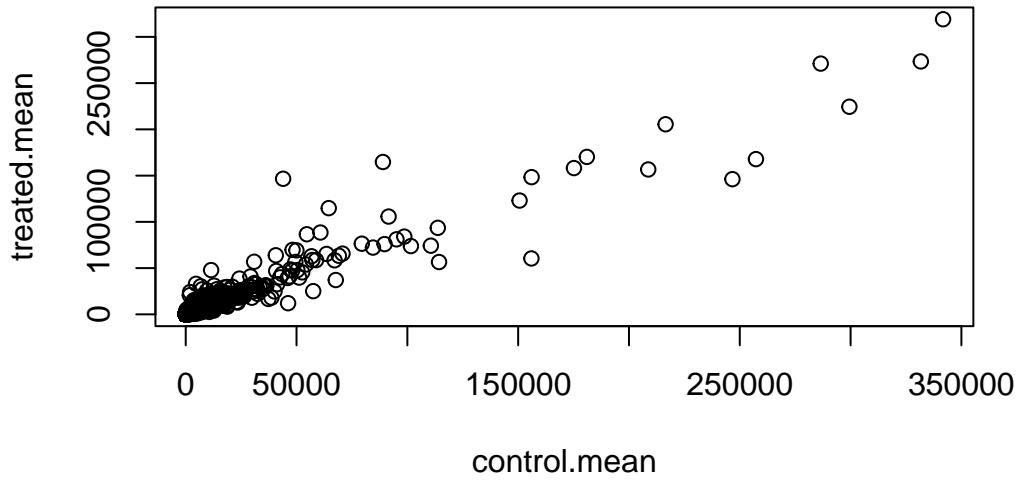
Now I have ‘control.mean’ and ‘treated.mean’. Let’s put them together for safe keeping & ease of use later by combining them.

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

	control.mean	treated.mean
ENSG000000000003	900.75	658.00
ENSG000000000005	0.00	0.00
ENSG000000000419	520.50	546.00
ENSG000000000457	339.75	316.50
ENSG000000000460	97.25	78.75
ENSG000000000938	0.75	0.00

Plotting things:

```
plot(meancounts)
```



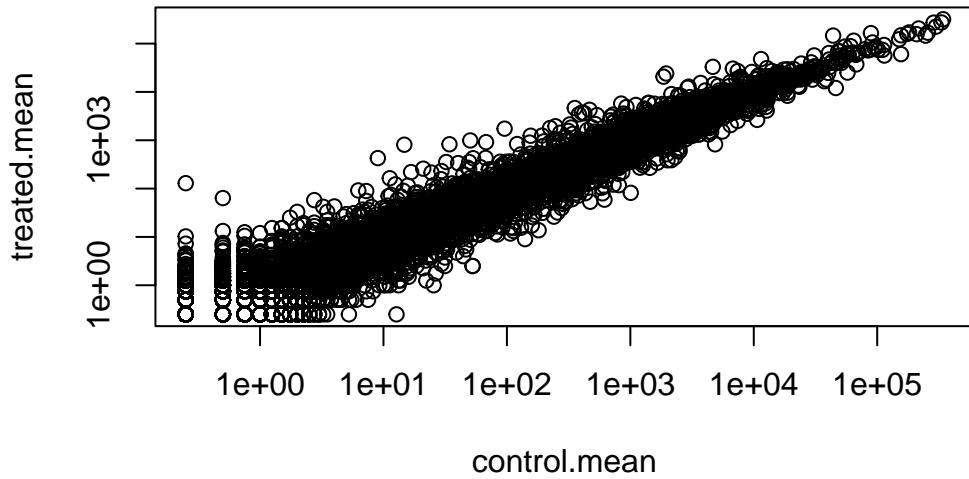
Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot? Q5: geom_point

This is very skewed and over a wide range, we use log to adjust Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this? A6: plot(meancounts, log="xy")

```
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 like working with log transformed data as it can help us make things more straightforward for interpretation If we have no change

```
log(20/20)
```

```
[1] 0
```

What about if we had a doubling?

```
log2(40/20)
```

```
[1] 1
```

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

```
[1] -1
```

We like working with log2 fold-change values. Let's calculate them for our data

```
meancounts$r2fc <- log2(meancounts$treated.mean/meancounts$control.mean)
head(meancounts)
```

	control.mean	treated.mean	r2fc
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

We want to filter out any genes (that is rows) where we have ZERO count data (Ex:ENSG000000000005)

```
to.keep inds <- rowSums(meancounts[,1:2] == 0) == 0
head(to.keep inds)
```

ENSG000000000003	TRUE	ENSG000000000005	FALSE	ENSG000000000419	TRUE	ENSG000000000457	TRUE	ENSG000000000460	TRUE
ENSG000000000938	FALSE								

```
mycounts <- meancounts[to.keep inds,]
nrow(mycounts)
```

```
[1] 21817
```

A common threshold for calling genes as differentially expressed is a log2 fold-change of +2 or -2.

```
sum(mycounts$log2fc >= +2)
```

```
[1] 0
```

```
= 314
```

What percent is this?

```
round((sum(mycounts$log2fc >= -2) / nrow(mycounts)) * 100, 2)
```

```
[1] 0
```

```
= 2.22
```

#4. DESeq2 analysis

We need some stats to check if the drug induced diff is significant! Let's turn to doing this the correct way with the DESeq2 package.

```
library(DESeq2)
```

Main function in the DESeq package is called 'deseq()'. It wants our count data & our colData (metadata) as input in a specific way.

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

```
converting counts to integer mode
```

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

Now I can run the DESeq analysis

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

```
results(dds)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
  baseMean log2FoldChange    lfcSE      stat     pvalue
  <numeric>      <numeric> <numeric> <numeric>
ENSG000000000003  747.1942   -0.3507030  0.168246 -2.084470 0.0371175
ENSG000000000005   0.0000      NA        NA        NA        NA
ENSG000000000419  520.1342   0.2061078  0.101059  2.039475 0.0414026
ENSG000000000457  322.6648   0.0245269  0.145145  0.168982 0.8658106
ENSG000000000460   87.6826   -0.1471420  0.257007 -0.572521 0.5669691
...
...
ENSG00000283115  0.000000      NA        NA        NA        NA
ENSG00000283116  0.000000      NA        NA        NA        NA
ENSG00000283119  0.000000      NA        NA        NA        NA
ENSG00000283120  0.974916   -0.668258  1.69456  -0.394354 0.693319
ENSG00000283123  0.000000      NA        NA        NA        NA
  padj
  <numeric>
ENSG000000000003  0.163035
ENSG000000000005   NA
ENSG000000000419  0.176032
ENSG000000000457  0.961694
ENSG000000000460  0.815849
...
...
ENSG00000283115   NA
ENSG00000283116   NA
ENSG00000283119   NA
ENSG00000283120   NA
ENSG00000283123   NA
```

Noe what we have got so far is the log2 fold-change and the adjusted p-value for the significance

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

```
head(res)
```

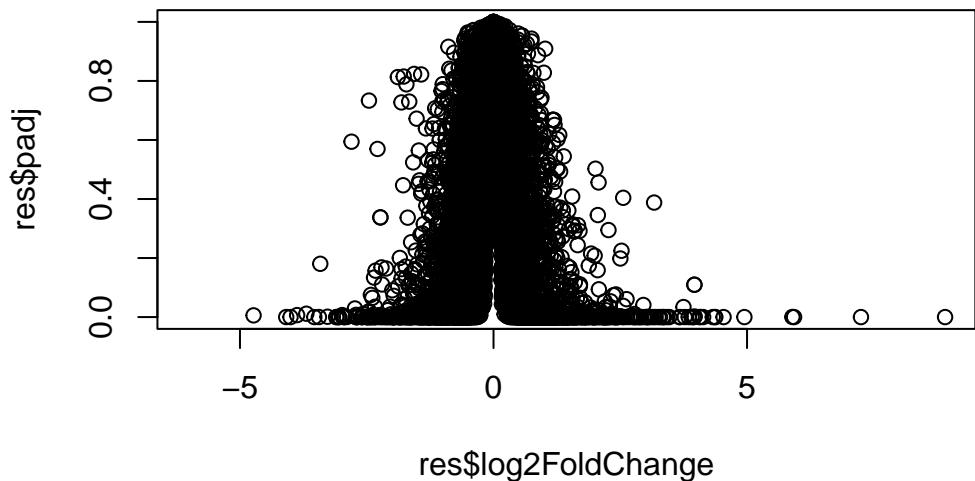
```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
```

```
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
```

#6. Data Visualization

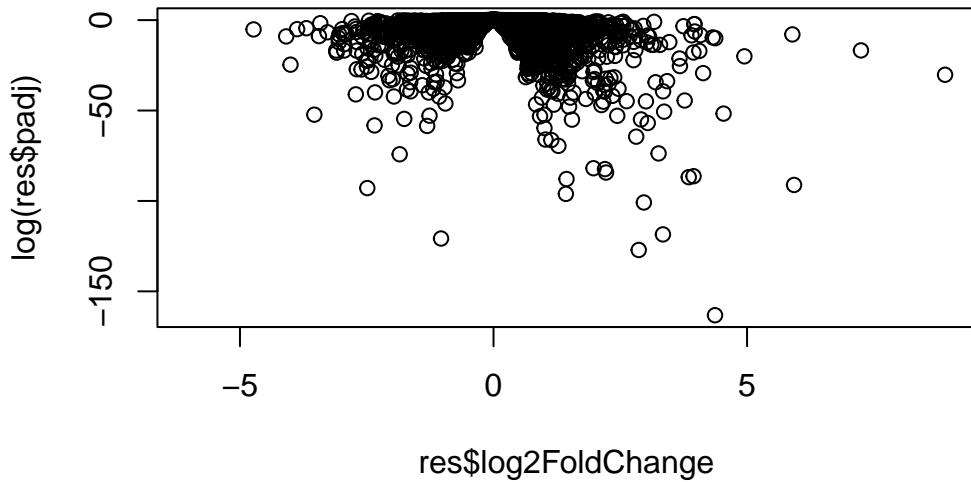
A First plot

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



^Plot sucked cause all P-values are down below zero. They're upsidedown like Australia

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

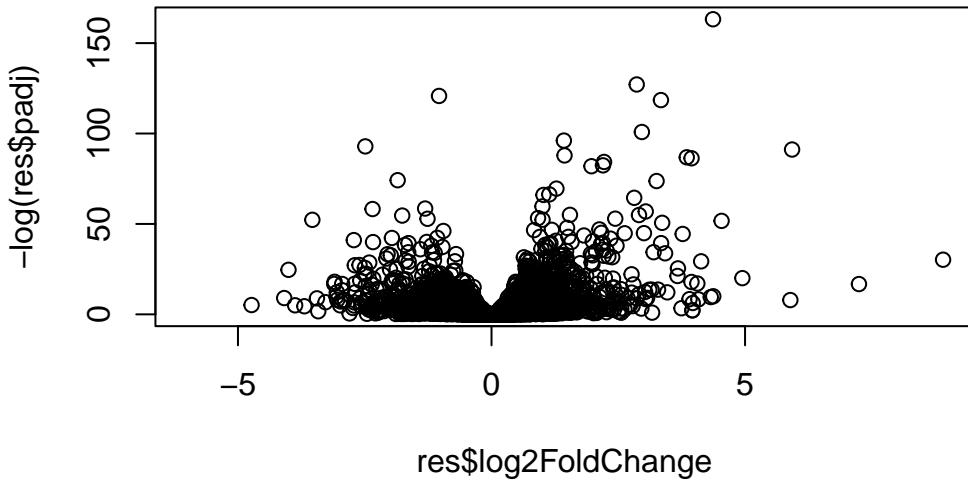


```
log(0.05)
```

```
[1] -2.995732
```

We can flip the y-axis so the plot doesn't look upside-down

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



```

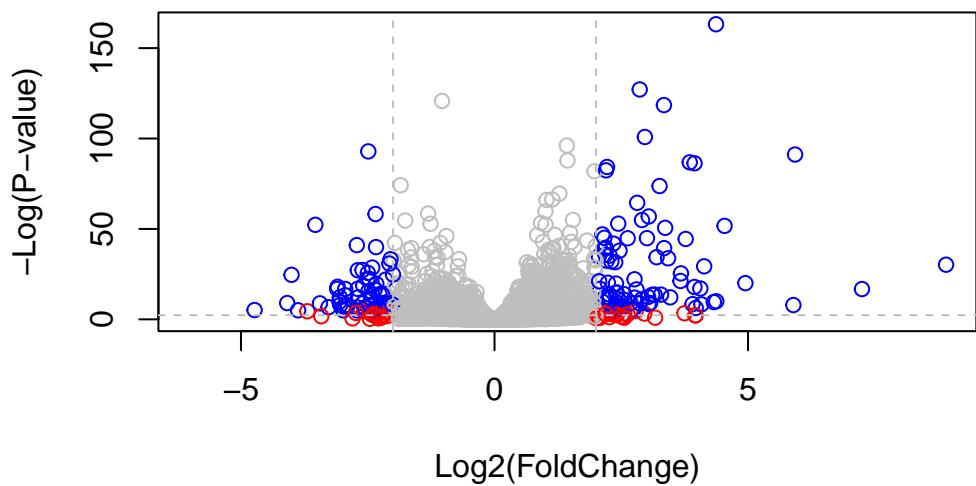
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

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

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

abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)

```



#7. Pathway analysis