Class 13: RNA Seq pt.1

Lily Huynh (PID: A16929651)

2025-02-18

Table of contents

Today we will analyze data from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

Import countData and colData

There are two data sets I need to import/read

- `countData` the transcript counts per gene (rows) in the different experiments
- `colData` information (a.k.a. metadata) about the columns (i.e. experiments) in `countData`.

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

We can have a peak at these with `head()`

```
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?

Ans. There are 38,694 genes in this dataset.

```
nrow(counts)
```

```
[1] 38694
```

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

Ans. There are 4 control cell lines.

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

```
control treated
      4       4
```

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

```
[1] 4
```

Toy differential gene expression

We can find the average (mean) count values per gene for all "control" experiments and compare it to the mean values for "treated".

- Extract all "control" columns from the `counts` data
- Find the mean value for each gene

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

```
dim(control.counts)
```

```
[1] 38694     4
```

Now find the row wise mean

```
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? Is there a function that could help here?
>
> Ans. I would use the `ncol()` function

```
control.mean <- rowSums(control.counts)/ncol(control.counts)
head(control.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

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)
head(treated.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         658.00            0.00          546.00          316.50           78.75
ENSG00000000938
           0.00
```

Let's put these two mean values together for easy book-keeping

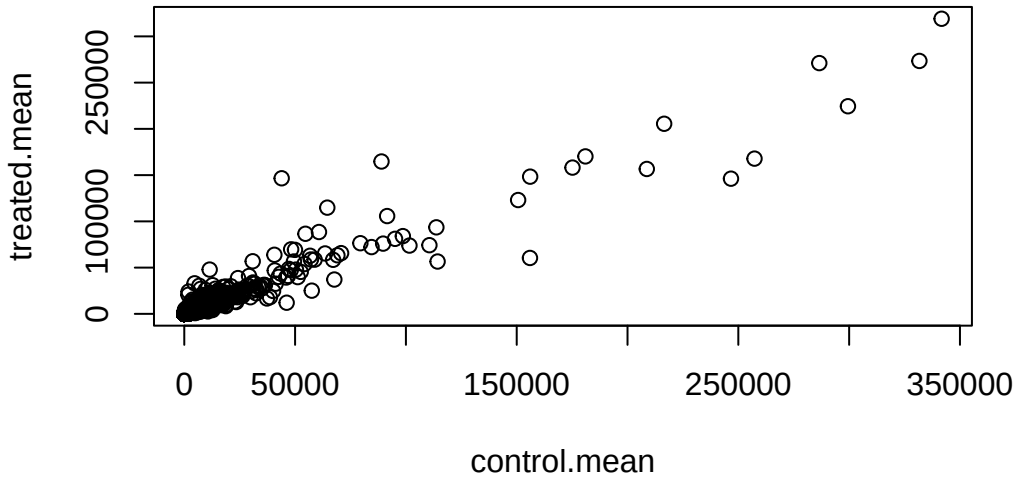
```
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)
```

```
                control.mean treated.mean
ENSG00000000003       900.75       658.00
ENSG00000000005         0.00         0.00
ENSG00000000419       520.50       546.00
ENSG00000000457       339.75       316.50
ENSG00000000460        97.25        78.75
ENSG00000000938         0.75         0.00
```

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.

Let's have a look - i.e. plot control.mean vs treated.mean

```
plot(meancounts)
```

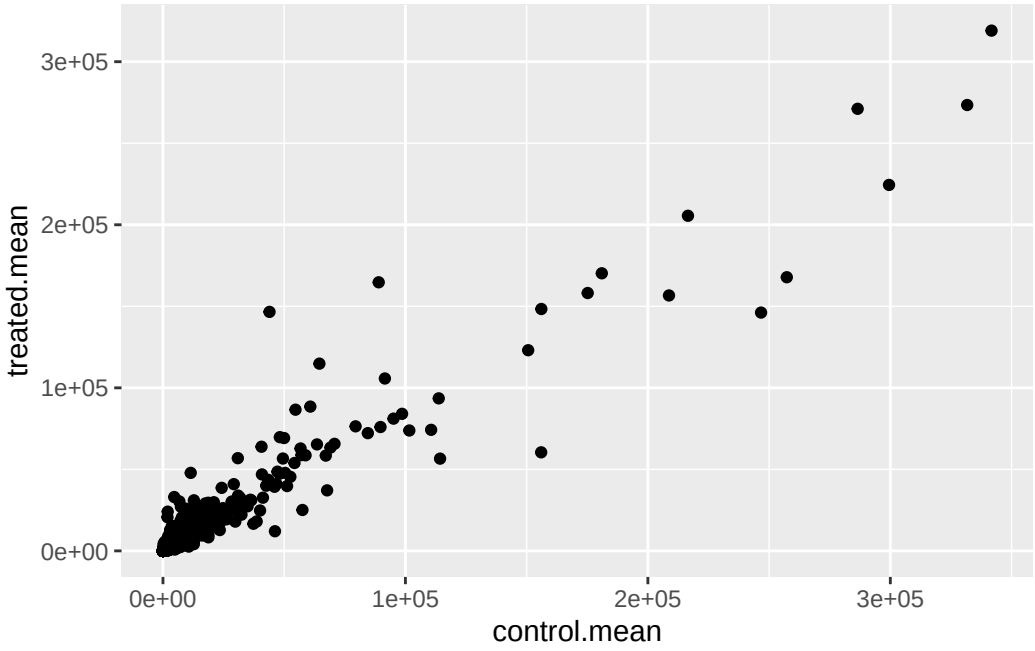


Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

Ans. I would use geom_point() function.

```
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?

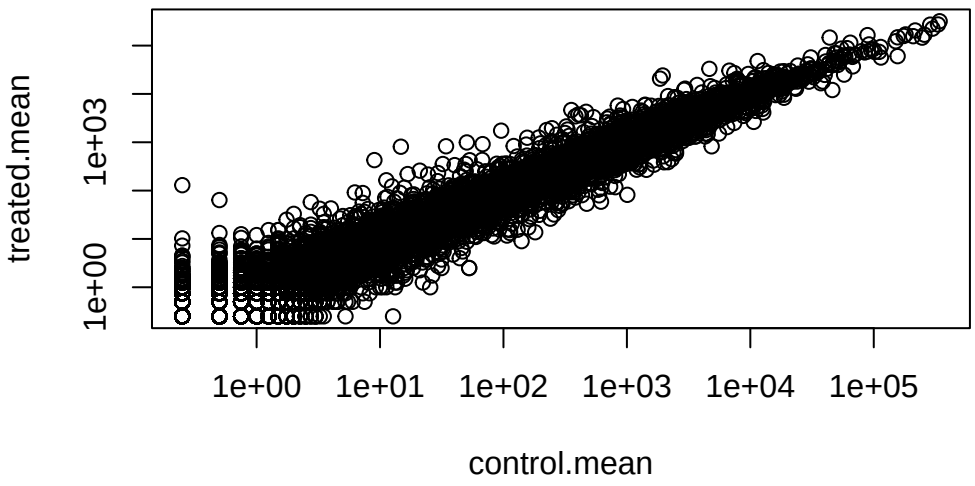
Ans. The log argument to plot().

Whenever we see data that is heavily skewed like this, we often log transform it so we can see what is going on more easily.

```
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 often work in log2 units as this makes the math easier. Let's have a play to see this

```
# treated / control
log2(20/20)
```

```
[1] 0
```

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

```
[1] 1
```

```
log2(80/20)
```

```
[1] 2
```

```
# treated/control
log2(20/40)
```

```
[1] -1
```

We can now add a "log2 fold-change" values to our `meancounts` data set.

```
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
```

```
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?
>
> Ans. The purpose of the arr.ind argument is to get rid of the NaN and infinity values from the table because they aren't considered TRUE values. The unique() function is used to make sure we don't count the rows twice when they have a 0.

We need to filter out zero count genes - i.e. remove the rows (genes) that have a 0 value in either control or treated means.

```
to.keep <- rowSums(meancounts[,1:2] == 0) ==0
mycounts2 <- meancounts[to.keep,]
nrow(mycounts2)
```

```
[1] 21817
```

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

Ans. There are 250 up regulated genes that have a greater than 2 fc level.

```
up.inds2 <- mycounts$log2fc > 2
sum(up.inds2)
```

```
[1] 250
```

How many genes are "up" regulated at the common log2 fold-change threshold of +2.

```
up.inds <- meancounts$log2fc>=2
sum(up.inds, na.rm = T)
```

```
[1] 1910
```

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

Ans. There are 367 down regulated genes that have a value greater than 2 fc level.

```
down.inds2 <- mycounts$log2fc < (-2)
sum(down.inds2)
```

```
[1] 367
```

How many genes are "down" regulated at the threshold of -2?

```
down.inds <- meancounts$log2fc <=-2
sum(down.inds, na.rm = T)
```

```
[1] 2330
```

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

Ans. I do not trust these results because we didn't calculate the p-value yet. Therefore, we don't know if the differences in the data is signficiant or not.

DESeq2 analysis

To do this the right way, we need to consider the significance of the differences not just their magnitude.

```
library(DESeq2)
```

To use this package, it wants countData and colData in a specific format.

```
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
```

```
dds2 <- DESeq(dds)
```

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

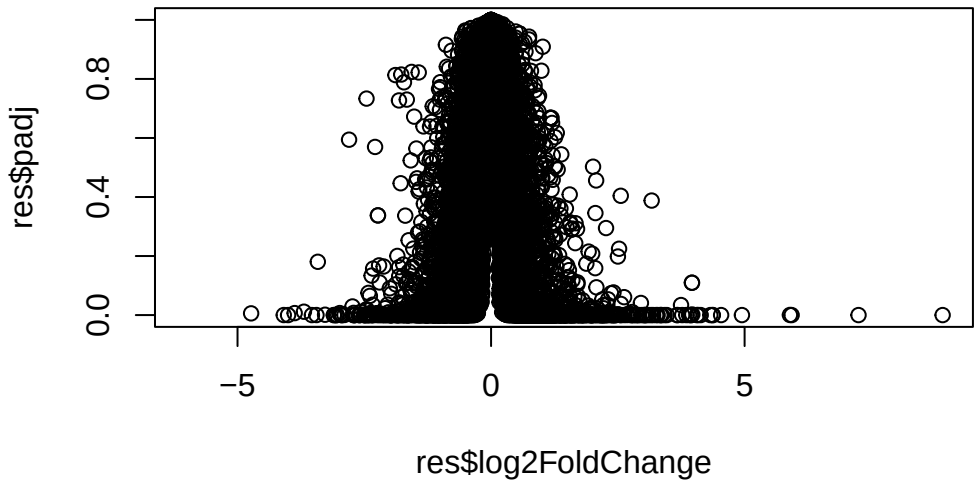
Extract my results

```
res <- results(dds2)
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
```

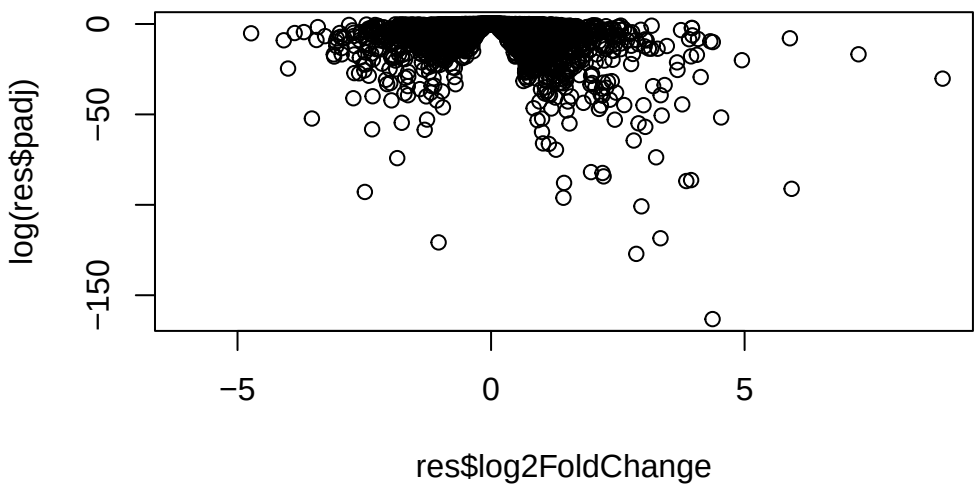
Plot of fold-change vs P-value (adjusted for multiple testing)

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



Take the log of the P-value

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




```
log(0.01)
```

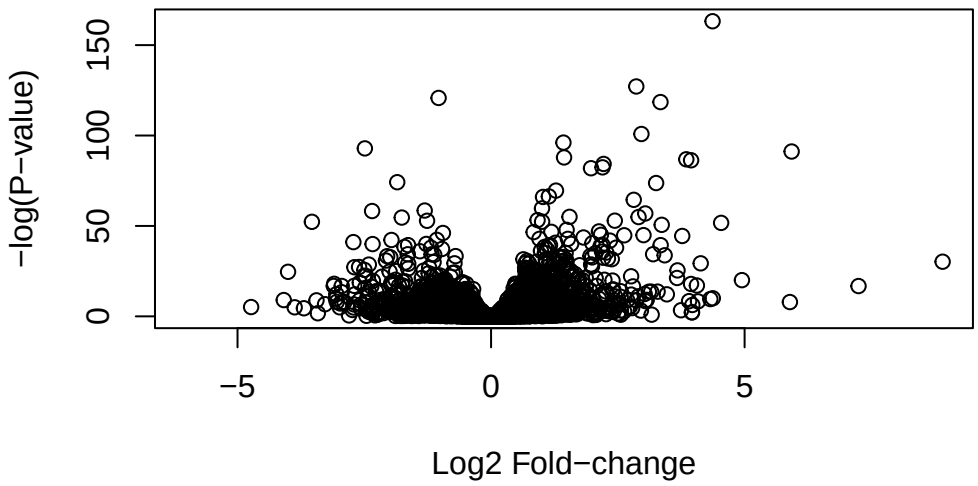
```
[1] -4.60517
```

```
log(0.0000000001)
```

```
[1] -23.02585
```

We can just flip that y-axis by putting a minus sign on it

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



Let's save our work to date

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

To finish off, let's make a nicer volcano plot

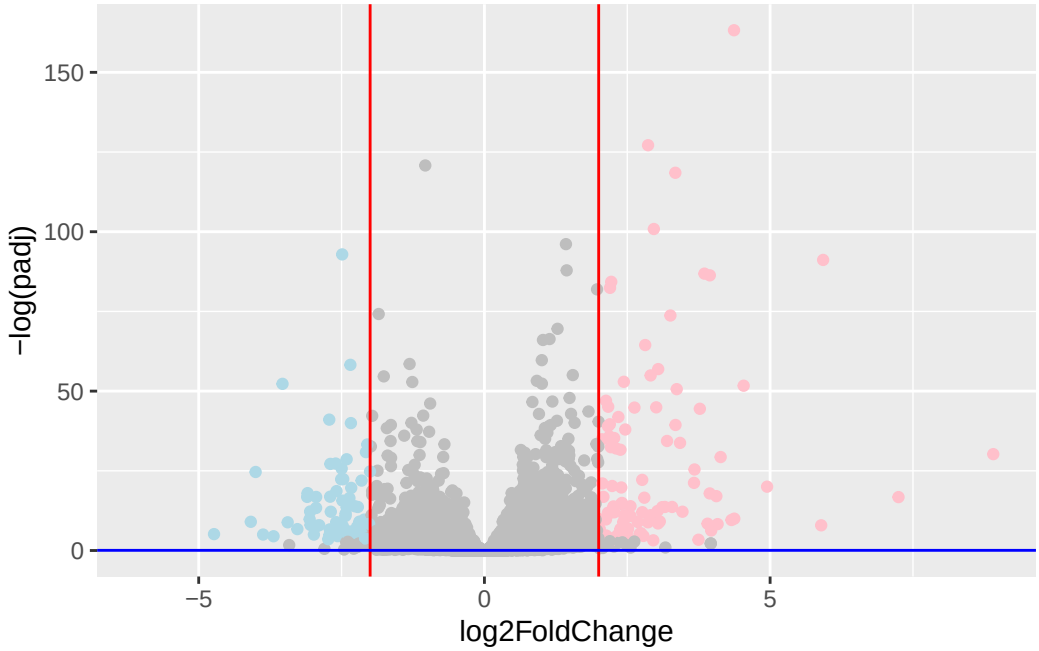
- Add the log2 threshold lines at +2/-2
- Add P-value threshold lines at 0.05
- Add color to highlight the subset of genes that meet both of the above thresholds.

Make it with ggplot please...

```
mycols <- rep("gray", nrow(res))
mycols[res$log2FoldChange>=2] <- "pink"
mycols[res$log2FoldChange <= -2] <- "lightblue"
mycols[res$padj > 0.05] <- "gray"
```

```
library(ggplot2)
ggplot(res) +
  aes(log2FoldChange, -log(padj)) +
  geom_point(col=mycols) +
  geom_vline(xintercept= c(-2,2), col="red") +
  geom_hline(yintercept=0.05, col="blue")
```

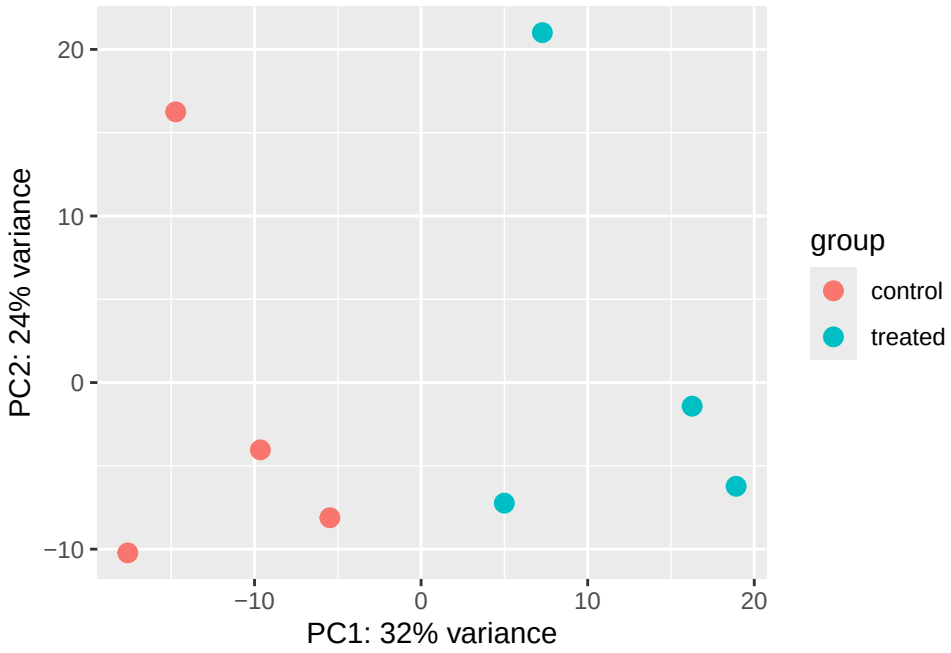
```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```



PCA

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