Class 16 HW

Kavi (PID: A69046927)

Table of contents

Principal Component Analysis 2
Differential Expression Analysis 7

Back on our laptop we can now use R and Bioconductor tools to further explore this large scale dataset.

For example there is an R function called tximport() in the tximport package, which enables straightforward import of Kallisto results

With each sample having its own directory containing the Kallisto output, we can import the transcript count estimates into R using:

```
library(tximport)
```

```
Warning: package 'tximport' was built under R version 4.5.2
```

```
# setup the folder and filenames to read
folders <- dir(pattern="SRR21568*")
samples <- sub("_quant", "", folders)
files <- file.path( folders, "abundance.h5" )
names(files) <- samples

txi.kallisto <- tximport(files, type = "kallisto", txOut = TRUE)
```

```
1 2 3 4
```

```
head(txi.kallisto$counts)
```

```
                SRR2156848 SRR2156849 SRR2156850 SRR2156851
ENST00000539570          0          0    0.00000          0
ENST00000576455          0          0    2.62037          0
ENST00000510508          0          0    0.00000          0
ENST00000474471          0          1    1.00000          0
ENST00000381700          0          0    0.00000          0
ENST00000445946          0          0    0.00000          0
```

We now have our estimated transcript counts for each sample in R. We can see how many transcripts we have for each sample, and how many transcripts are detected in at least one sample:

```
colSums(txi.kallisto$counts)
```

```
SRR2156848 SRR2156849 SRR2156850 SRR2156851
   2563611    2600800    2372309    2111474
```

```
sum(rowSums(txi.kallisto$counts)>0)
```

```
[1] 94561
```

Before subsequent analysis, we might want to filter out those annotated transcripts with no reads:

```
to.keep <- rowSums(txi.kallisto$counts) > 0
kset.nonzero <- txi.kallisto$counts[to.keep,]
```

And those with no change over the samples:

```
keep2 <- apply(kset.nonzero,1,sd)>0
x <- kset.nonzero[keep2,]
```

Principal Component Analysis

We can now apply any exploratory analysis technique to this counts matrix. As an example, we will perform a PCA of the transcriptomic profiles of these samples.

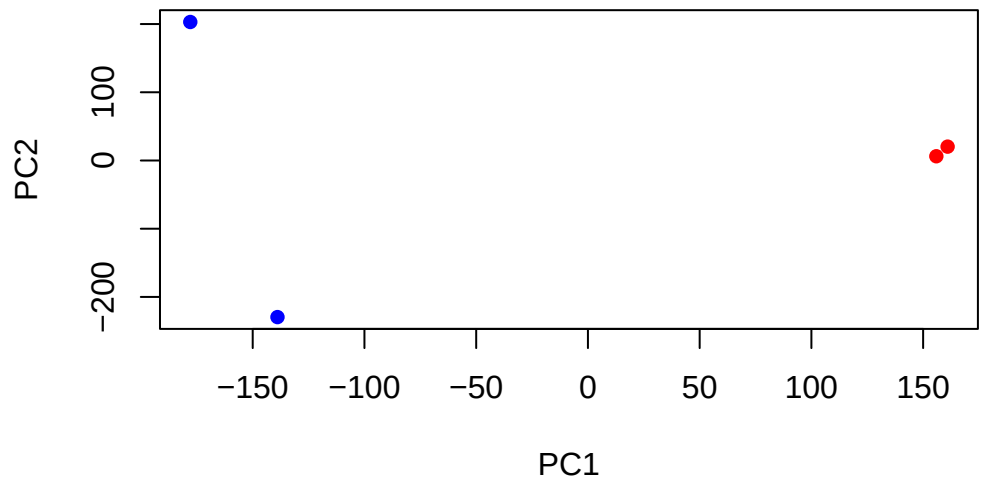
Now we compute the principal components, centering and scaling each transcript's measured levels so that each feature contributes equally to the PCA:

```
pca <- prcomp(t(x), scale=TRUE)
summary(pca)
```

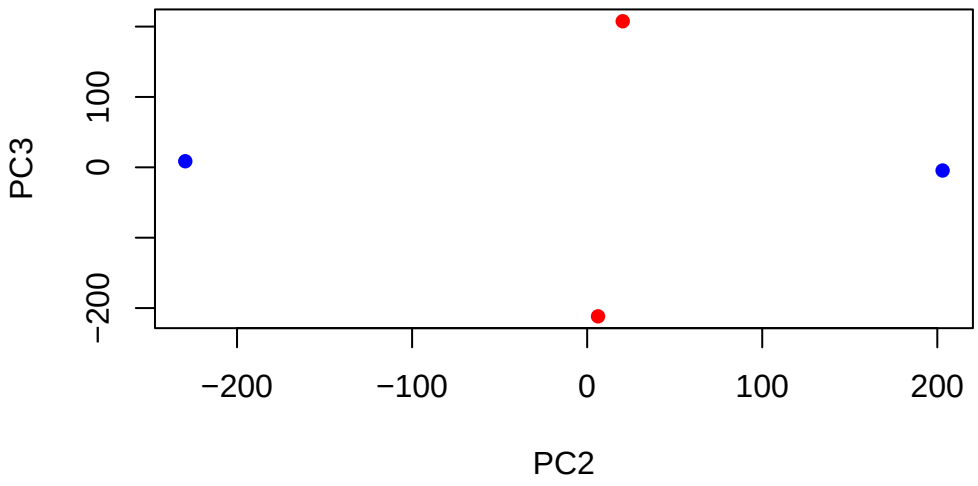
```
Importance of components:
                            PC1      PC2      PC3   PC4
Standard deviation     183.6379 177.3605 171.3020 1e+00
Proportion of Variance   0.3568   0.3328   0.3104 1e-05
Cumulative Proportion    0.3568   0.6895   1.0000 1e+00
```

Now we can use the first two principal components as a co-ordinate system for visualizing the summarized transcriptomic profiles of each sample:

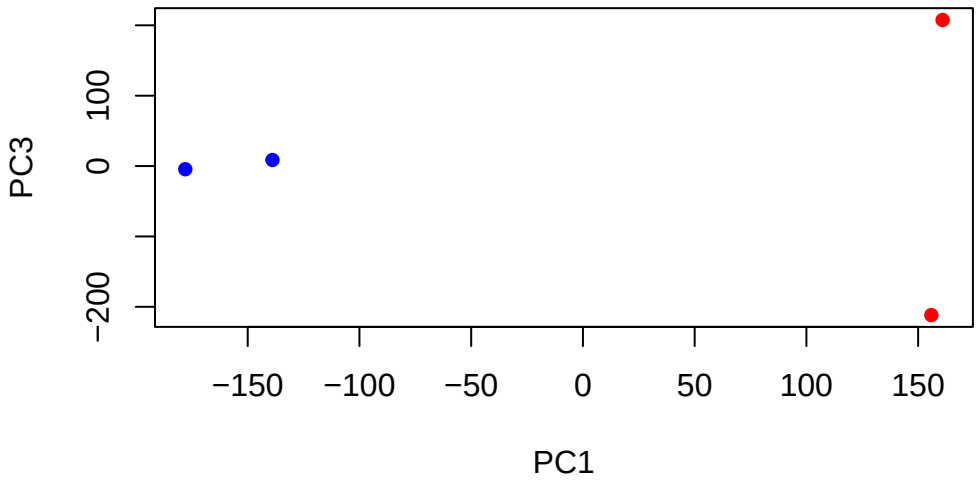
```
plot(pca$x[,1], pca$x[,2],
     col=c("blue","blue","red","red"),
     xlab="PC1", ylab="PC2", pch=16)
```




```
plot(pca$x[,2], pca$x[,3],
     col=c("blue","blue","red","red"),
     xlab="PC2", ylab="PC3", pch=16)
```



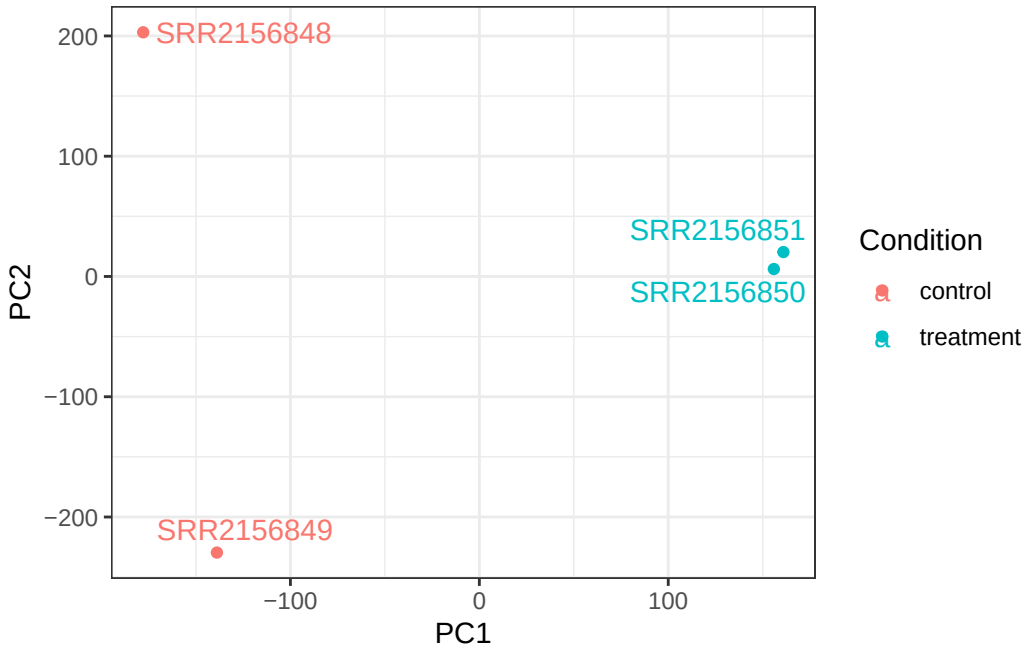

```
plot(pca$x[,1], pca$x[,3],
     col=c("blue","blue","red","red"),
     xlab="PC1", ylab="PC3", pch=16)
```



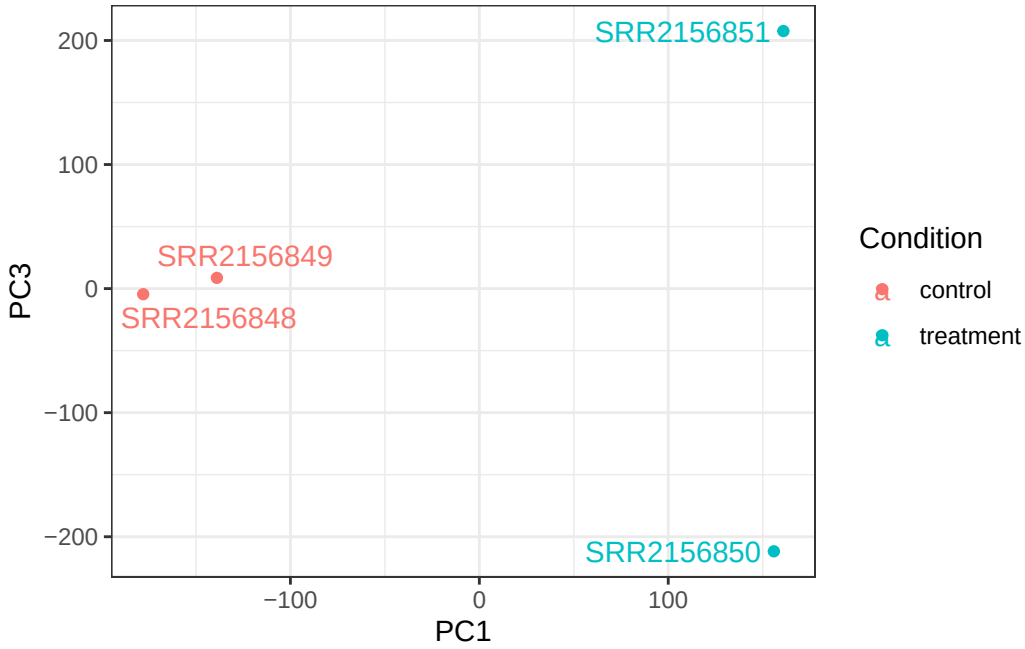
```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.5.2
```

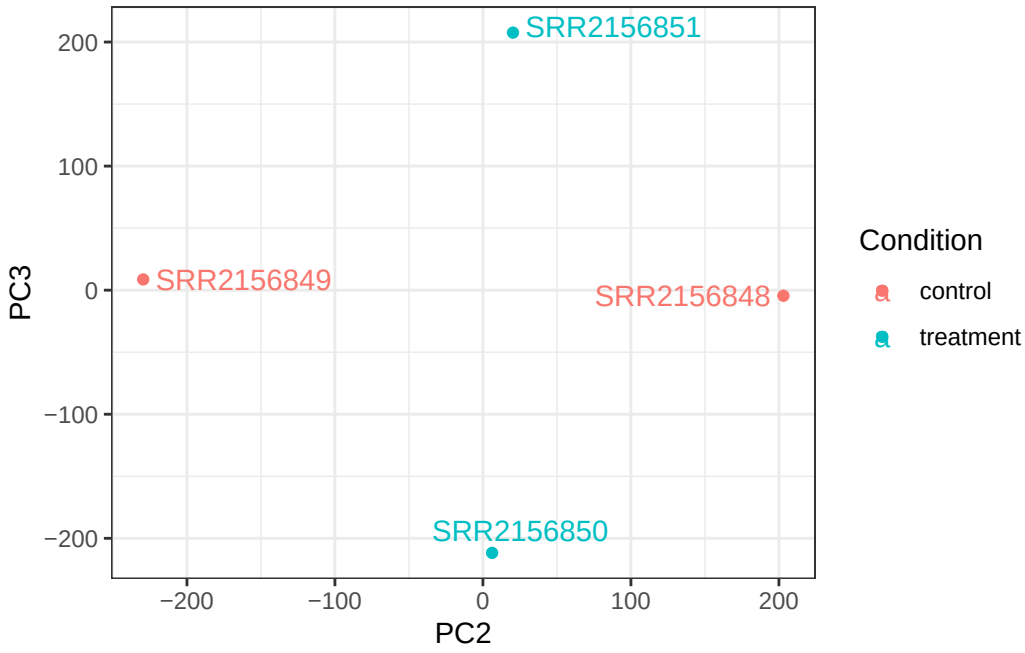
```
library(ggrepel)

# Make metadata object for the samples
colData <- data.frame(condition = factor(rep(c("control", "treatment"), each = 2)))
rownames(colData) <- colnames(txi.kallisto$counts)

# Make the data.frame for ggplot
y <- as.data.frame(pca$x)
y$Condition <- as.factor(colData$condition)

ggplot(y) +
  aes(PC1, PC2, col=Condition) +
  geom_point() +
  geom_text_repel(label=rownames(y)) +
  theme_bw()
```



```
ggplot(y) +
  aes(PC1, PC3, col=Condition) +
  geom_point() +
  geom_text_repel(label=rownames(y)) +
  theme_bw()
```




```
ggplot(y) +
  aes(PC2, PC3, col=Condition) +
  geom_point() +
  geom_text_repel(label=rownames(y)) +
  theme_bw()
```



Differential Expression Analysis

We can use DESeq2 to complete the differential-expression analysis that we are already familiar with:

An example of creating a DESeqDataSet for use with DESeq2:

```
library(DESeq2)
```

```
Warning: package 'DESeq2' was built under R version 4.5.2
```

```
Warning: package 'matrixStats' was built under R version 4.5.2
```

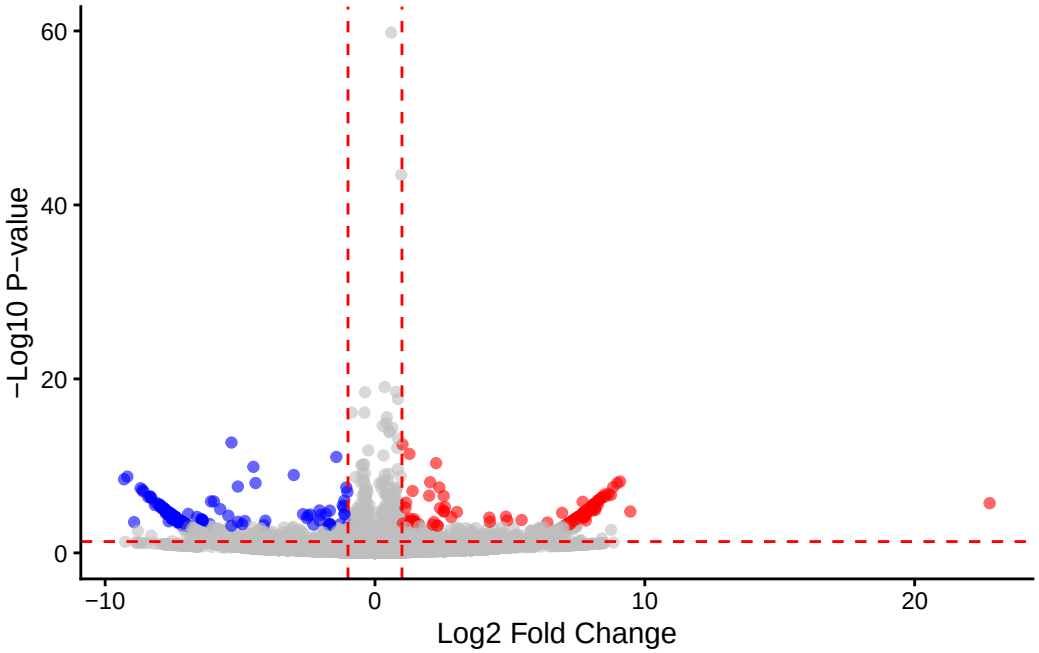
```
sampleTable <- data.frame(condition = factor(rep(c("control", "treatment"), each = 2)))
rownames(sampleTable) <- colnames(txi.kallisto$counts)
dds <- DESeqDataSetFromTximport(txi.kallisto,
                                sampleTable,
                                ~condition)
dds <- DESeq(dds)
res <- results(dds)
head(res)
```

```
log2 fold change (MLE): condition treatment vs control
Wald test p-value: condition treatment vs control
DataFrame with 6 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENST00000539570  0.000000             NA        NA        NA        NA
ENST00000576455  0.761453       3.155061   4.86052 0.6491203  0.516261
ENST00000510508  0.000000             NA        NA        NA        NA
ENST00000474471  0.484938       0.181923   4.24871 0.0428185  0.965846
ENST00000381700  0.000000             NA        NA        NA        NA
ENST00000445946  0.000000             NA        NA        NA        NA
                     padj
                <numeric>
ENST00000539570        NA
ENST00000576455        NA
ENST00000510508        NA
ENST00000474471        NA
ENST00000381700        NA
ENST00000445946        NA
```

```r
my_cols <- rep("gray", nrow(res))
my_cols[which(res$padj < 0.05 & res$log2FoldChange > 1)] <- "red"
my_cols[which(res$padj < 0.05 & res$log2FoldChange < -1)] <- "blue"
```

```r
library(ggplot2)
library(ggrepel)

ggplot(res, aes(x = log2FoldChange, y = -log10(pvalue))) +
  geom_point(col = my_cols, alpha = 0.6) +
  theme_classic() +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "red") +
  geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "red") +
  labs(x = "Log2 Fold Change", y = "-Log10 P-value")
```

```
Warning: Removed 89410 rows containing missing values or values outside the scale range
(`geom_point()`).
```




```
  # Only label significant hits:
  geom_text_repel(
    data = subset(res, padj < 0.05 & abs(log2FoldChange) > 1),
    aes(label = genename),
    size = 3
  )
```

```
mapping: label = ~genename
geom_text_repel: parse = FALSE, na.rm = FALSE, box.padding = 0.25, point.padding = 1e-06, mi
stat_identity: na.rm = FALSE
position_identity
```