Methun George

1/18/2020

DIFFERENTIAL EXPRESSION ANALYSIS

Differential expression analysis means taking the normalised read count data and performing statistical analysis to discover quantitative changes in expression levels between experimental groups. Load Featurecount table into R.

```
  countdata<- read.table
("/home/mlsi/RNASeq/countTable/featureCounts.txt",header=TRUE, row.names=1)
  class (countdata)

## [1] "data.frame"
```

Edit Featurecount Table

In this step we edit the data by deleting some unwanted data or by adding some data that we need to display in the finel result.

```
countdata <- countdata[ ,6:ncol(countdata)]
```

To remove .bam or -.sam from filenames we uses the following commands.

```
colnames(countdata) <- gsub
("\\X.home.mlsi.RNASeq.mapping.","",colnames(countdata))
colnames(countdata) <- gsub ("\\.UHR_[123].bam","",colnames(countdata))
colnames(countdata) <- gsub ("\\.HBR_[123].bam","",colnames(countdata))
colnames(countdata)

## [1] "HBR_1" "HBR_2" "HBR_3" "UHR_1" "UHR_2" "UHR_3"
```

Next we need to convert the data.frame into a matrix and to do so we uses the followiong command.

```
countdata <- as.matrix(countdata)
class (countdata)

## [1] "matrix"
```

Design coldata

In coldata design we define all the characteristics of the sample which we uses and the characteristics we want to compare by performing R.

```
group<- factor(c(rep("HBR",3), rep("UHR",3)))
con<- factor(c(rep("cancer",3), rep("ctrl",3)))
```

Create a coldata frame.

Creating coldata frame is a part of coldata design. coldata frame makes sure that all the needed data slots are consistently provided and fulfill the requirements.

```
  coldata <- data.frame(row.names=colnames(countdata), group, con)
```

Colors for plots

we can select the colours for the upcoming plot. Moreover, we can select multiple colours accoding to our wishes.

```
library(RColorBrewer)
mycols <- brewer.pal(11, "Set3")[1:length(unique(group))]
```

Create DESeqDataSet

DESeqDataSet is a subclass of Ranged Summarized Experiment, used to store the input values, intermediate calculations and results of an analysis of differential expression. The constructor functions create a DESeqDataSet object from various types of input: a Ranged Summarized Experiment, a matrix, count files generated by the python package HTSeq, or a list from the tximport function in the tximport package.

```
library(DESeq2)

dds<- DESeqDataSetFromMatrix (countData= countdata, colData=coldata, design=
~ con)
```

Now we can take a look into the counts.

```
head(counts(dds))

##              HBR_1 HBR_2 HBR_3 UHR_1 UHR_2 UHR_3
## U2               0     0     0     0     0     0
## FRG1FP           0     0     0     0     0     0
## CU104787.1       0     0     0     0     0     0
## BAGE5            0     0     0     0     0     0
## ACTR3BP6         0     0     0     0     0     0
## 5_8S_rRNA        0     0     0     0     0     0
```

To Check the design of the DESEeqDataSets.

```
design(dds)

## ~con
```

Create the estimate Size Factors.

Which means that to create a data table with read counts normalized to library size, we should run the following commands.

```
dds <- estimateSizeFactors(dds)
sF<-sizeFactors(dds)
dds_norm_size_factor<- counts(dds, normalized=TRUE)
head(dds_norm_size_factor)
```

```
##              HBR_1 HBR_2 HBR_3 UHR_1 UHR_2 UHR_3
## U2               0     0     0     0     0     0
## FRG1FP           0     0     0     0     0     0
## CU104787.1       0     0     0     0     0     0
## BAGE5            0     0     0     0     0     0
## ACTR3BP6         0     0     0     0     0     0
## 5_8S_rRNA        0     0     0     0     0     0

write.table (dds_norm_size_factor, file =
"/home/mlsi/RNASeq/analysis/DESeq2/ddsNormSF.txt", sep = " ", col.names=NA)
```

Pre-Filtering

In pre-filtering step we filter out the lowly expressed genes and check the dimension of the dataset before further processing.

```
dds<- dds [rowSums(counts(dds)) > 1, ]
dim(dds)

## [1] 992   6
```

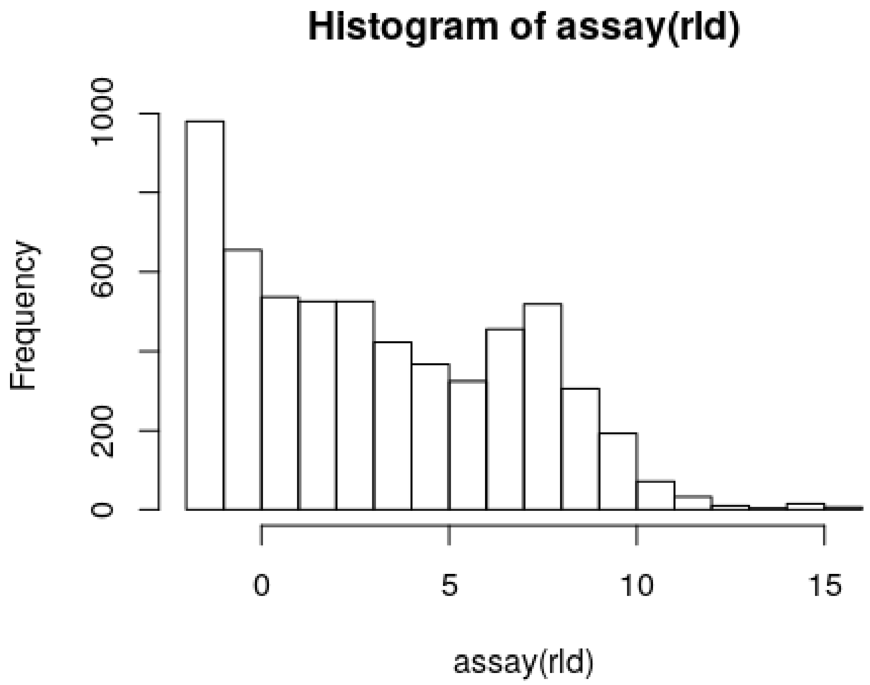
Rlog Transformation

In Rlog transformation we use the regularized log transformation for clustering/heatmaps of Count Data.

```
rld<- rlogTransformation(dds)
head(assay(rld))

##                      HBR_1      HBR_2      HBR_3      UHR_1      UHR_2
## LA16c-60D12.1 -0.4166441 -0.4389221 -0.4247142  0.6349742  0.6205201
## LA16c-60D12.2 -0.4904459 -0.5182887 -0.5004650 -0.5680166  1.2020601
## ZNF72P        -1.5452155 -1.5697664 -1.5539134 -1.6178908 -1.1229702
## BNIP3P2       -1.5452155 -1.5697664 -1.5539134 -1.6178908 -1.1229702
## LA16c-60G3.6  -1.5452155 -1.5697664 -1.5539134 -1.6178908 -1.1229702
## ARHGAP42P3    -1.1915067 -1.2311987 -1.2055688 -1.2804190 -0.2088327
##                      UHR_3
## LA16c-60D12.1 -0.4559831
## LA16c-60D12.2  0.1420802
## ZNF72P        -1.5899494
## BNIP3P2       -1.5899494
## LA16c-60G3.6  -1.5899494
## ARHGAP42P3    -1.2638289

hist(assay(rld))
```



Differential Expression Analysis via DESeq2

The package DESeq2 provides methods to test for differential expression by use of negative binomial generalized linear models; the estimates of dispersion and logarithmic fold changes incorporate data-driven prior distributions.

the design of the current DESeqDataSet(dds)

```
design(dds)

## ~con
```

Now we can run the DESeq-Pipeline for the current condition.

```
dds_con <- DESeq(dds)

## using pre-existing size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

Check factor setting and possible comparisons:

```
resultsNames(dds_con)
```

```
## [1] "Intercept"          "con_ctrl_vs_cancer"
```

Create results tables:

while creating the tables we can define specific contrast settings if you have more possible comparisons.

```
res_con<- results(dds_con, contrast=c("con", "cancer", "ctrl"))
```

A DESeqDataSet object must have an associated design formula. The design formula expresses the variables which will be used in modeling. The formula should be a tilde (~) followed by the variables with plus signs between them (it will be coerced into an formula if it is not already). The design can be changed later. However, then all differential analysis steps should be repeated, as the design formula is used to estimate the dispersions and to estimate the log2 fold changes of the model.

You can run the DESeq-Pipeline and results-function with different designs, depending on your defined coldata.

Explore the result tables:

```
summary(res_con)

## 
## out of 992 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)     : 182, 18%
## LFC < 0 (down)   : 199, 20%
## outliers [1]     : 0, 0%
## low counts [2]   : 250, 25%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results

mcols(res_con, use.names = TRUE)

## DataFrame with 6 rows and 2 columns
##                        type                                description
##                 <character>                                <character>
## baseMean       intermediate  mean of normalized counts for all samples
## log2FoldChange      results log2 fold change (MLE): con cancer vs ctrl
## lfcSE               results         standard error: con cancer vs ctrl
## stat                results         Wald statistic: con cancer vs ctrl
## pvalue              results      Wald test p-value: con cancer vs ctrl
## padj                results                       BH adjusted p-values
```

Now we can change the design of the DESeqDataSet and start a new analysis:

```
design (dds)<- ~group

design(dds)

## ~group
```

```
dds_group <- DESeq(dds)

## using pre-existing size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

resultsNames(dds_group)

## [1] "Intercept"        "group_UHR_vs_HBR"

res_group <- results(dds_group)
```

Exploring and exporting results

Shrinkage

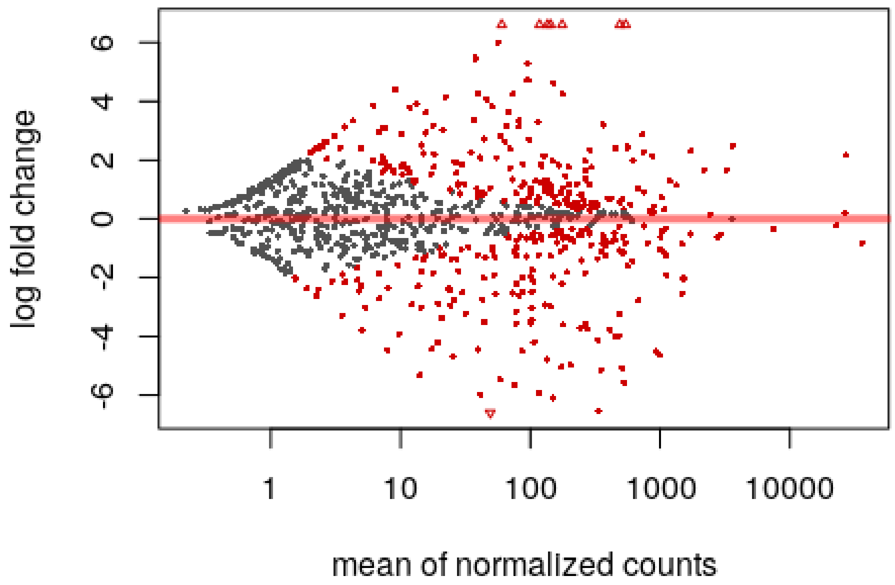
The shrinkage effect is useful for the visualisation as well as the ranking of different genes. It is more useful in visualization of the MA plot for the shrunken log2 fold changes. This removes the noice associated with log2 fold changes from low count genes without requiring arbitrary filtering thresholds. (coef–> check in resultsNames(dds))

```
resLFC_con<- lfcShrink(dds_con,coef=2)
```

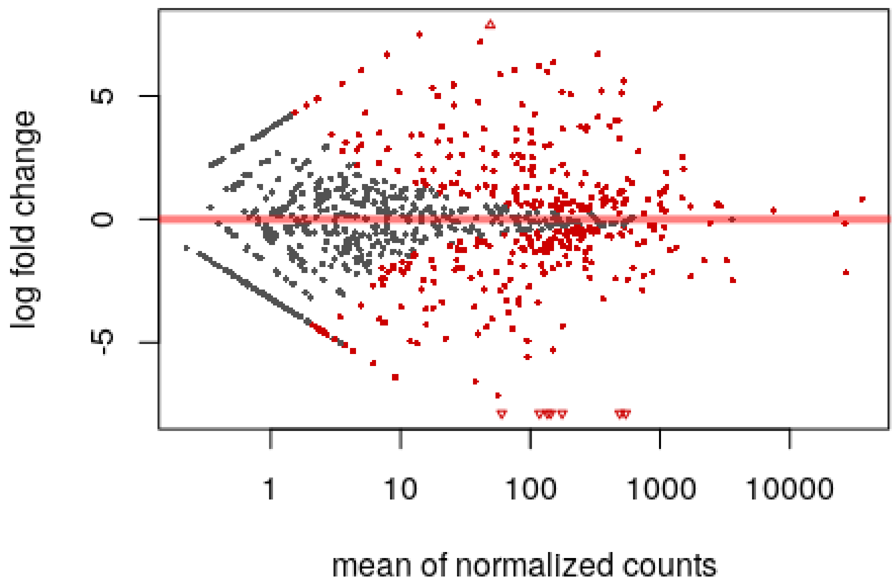
plotMA

A simple helper function that makes a so-called "MA-plot", i.e. a scatter plot of log2 fold changes (on the y-axis) versus the mean of normalized counts (on the x-axis). Here, the function plotMA shows the log2 fold changes attributable to a given variable over the mean of normalized counts for all the samples in the DESeqDataSet. Points will be colored red if the adjusted p value is less than 0.1. Points which fall out of the window are plotted as open triangles pointing either up or down.

```
plotMA(resLFC_con)
```

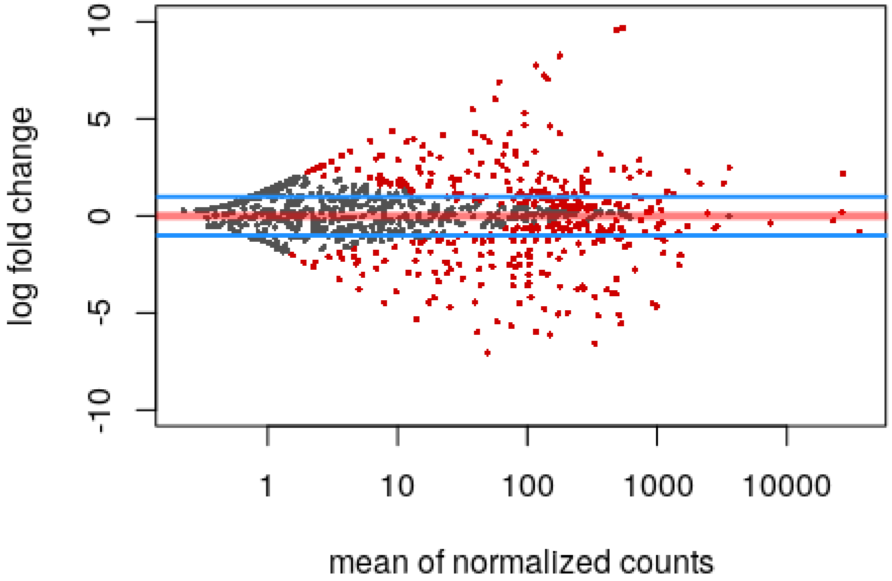


```r
plotMA(res_con)
```



Abline: h : the y-value(s) for horizontal line(s) [v : the x-value(s) for vertical line(s)] a, b : single values specifying the intercept and the slope of the line

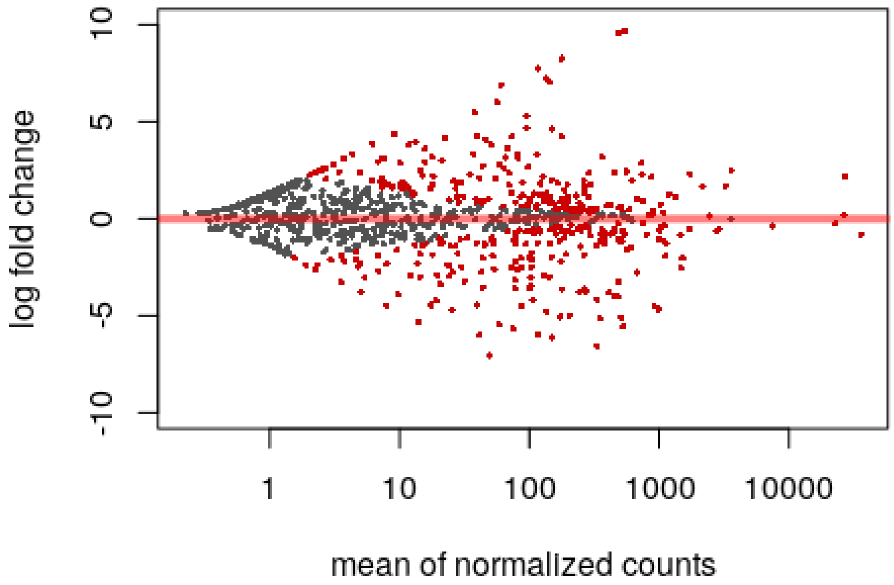
```
plotMA(resLFC_con, ylim=c(-10,10))
abline (h=c(-1,1), col="dodgerblue", lwd=2)
```



Interactively detect the row number of individual genes by clicking on the plot: (run plotMA and identify in console)

```
plotMA(resLFC_con, ylim=c(-10,10))

idx <- identify(resLFC_con$baseMean, resLFC_con$log2FoldChange)
```

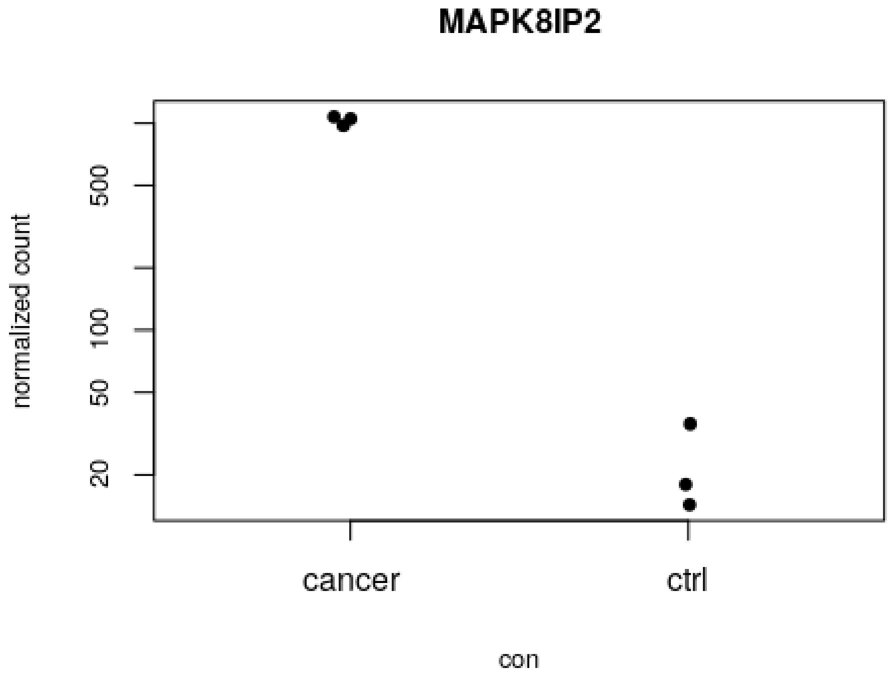


Here we can choose the point of interest just by clicking on the dots.

```
rownames(resLFC_con)[idx]
```

Visualize counts of a single gene of interest via plotCounts:

```
plotCounts(dds_con,gene="MAPK8IP2", intgroup="con", xlab="con",
cex=0.8, pch=19, cex.lab=0.8, cex.sub=0.8, cex.axis=0.8, cex.main=1)
```



Sample distance heatmap

The heatmap is another way to visualize hierarchical clustering. It's also called a false colored image, where data values are transformed to color scale. A heatmap of this distance matrix gives us an overview over similarities and dissimilarities between samples. Multiple versions possible:

Example 1:

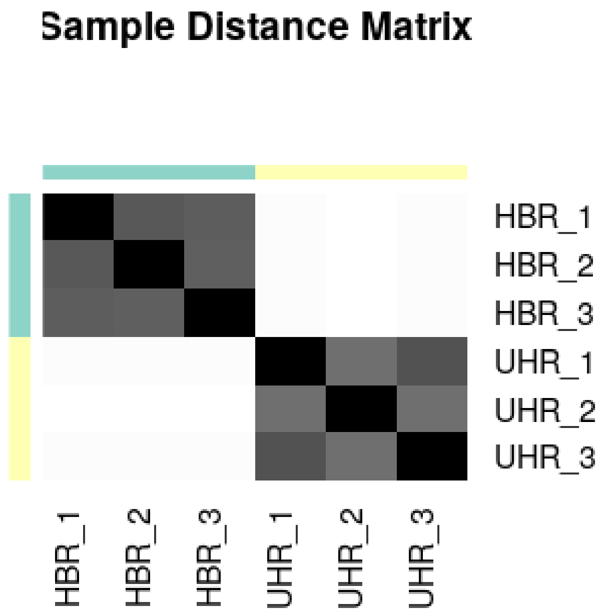
Convert regulized log transferred count data into a sample-dist-matrix:

```
sampleDists <- as.matrix(dist(t(assay(rld))))
```

Create the plot:

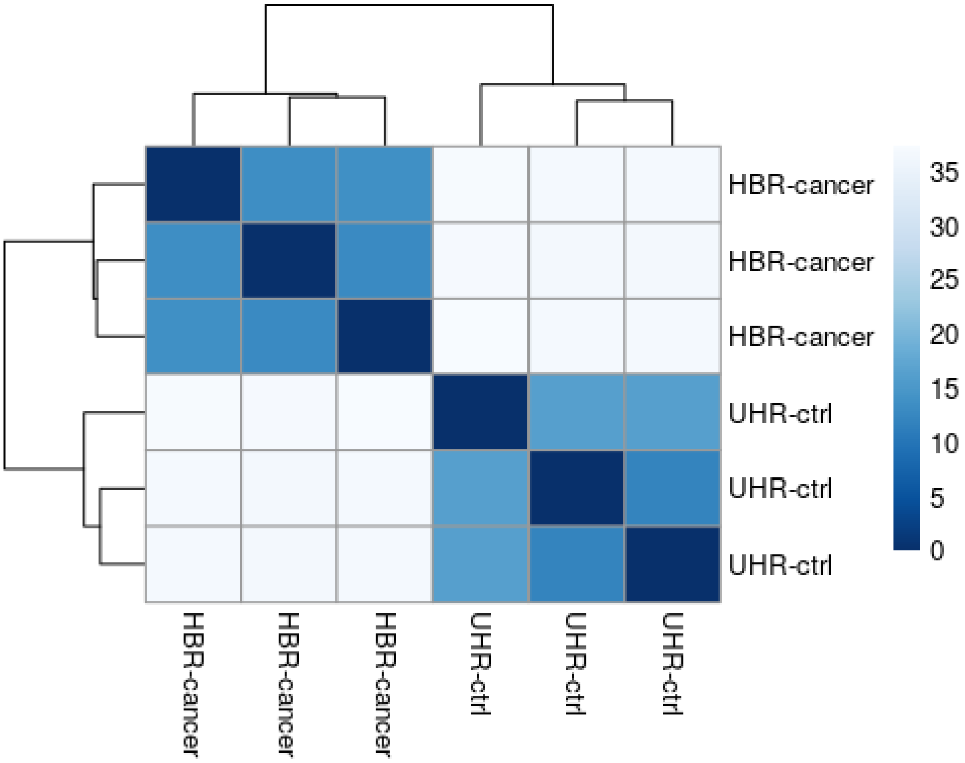
```
library("gplots")

heatmap.2(as.matrix(sampleDists), key=F, trace="none",Colv = c("cancer",
"ctrl"),Rowv =
"Colv",dendrogram= "none",
        col=colorpanel(100, "black", "white"),
        ColSideColors=mycols[con], RowSideColors=mycols[con],
        margin=c(10, 10), main="Sample Distance Matrix")
```



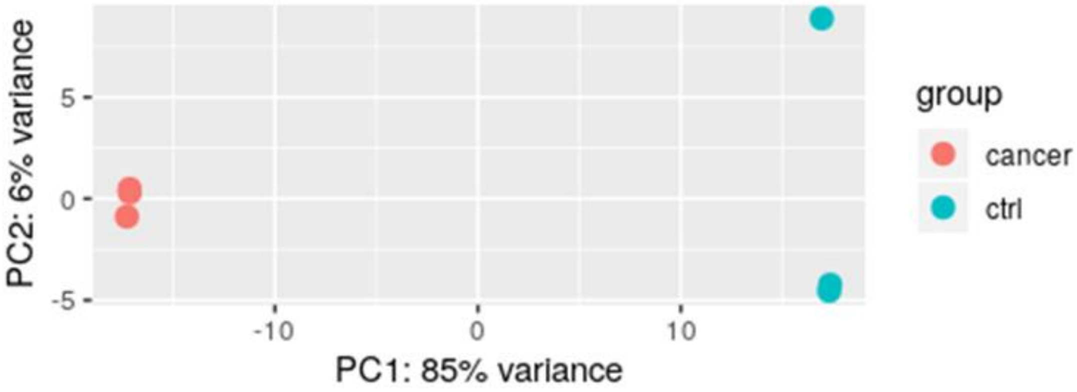
Example 2:

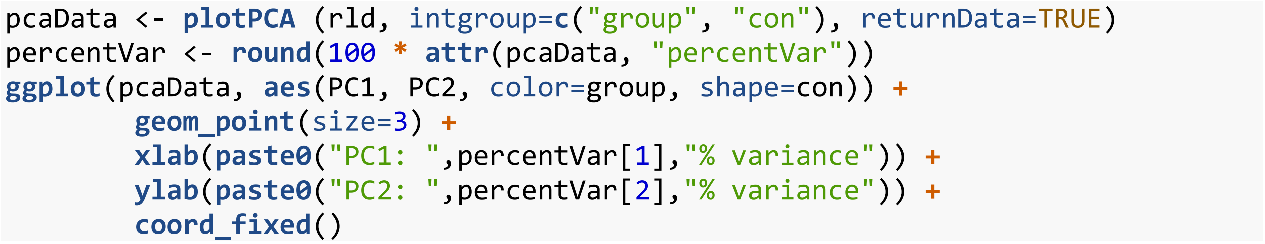
```
library("RColorBrewer")
sampleDists <- dist(t(assay(rld)))
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rld$group, rld$con, sep="-")
colnames(sampleDistMatrix) <- paste(rld$group, rld$con, sep="-")
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
library("pheatmap")
pheatmap(sampleDistMatrix,clustering_distance_rows=sampleDists,
clustering_distance_cols=sampleDists,col=colors)
```



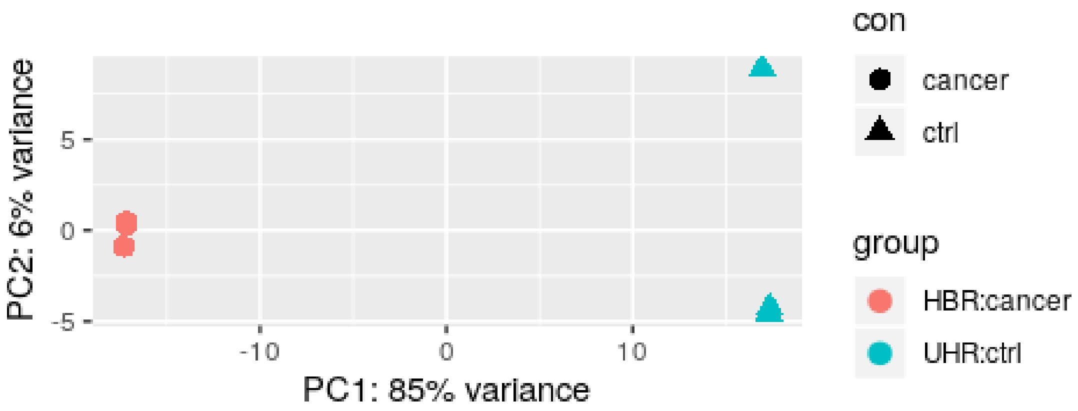
Principal component plot DESeq uses negative binomial distribution. Such distributions have two parameters: mean and dispersion. The dispersion is a parameter describing how much the variance deviates from the mean.

```
plotDispEsts(dds_con, main="Dispersion plot")
```



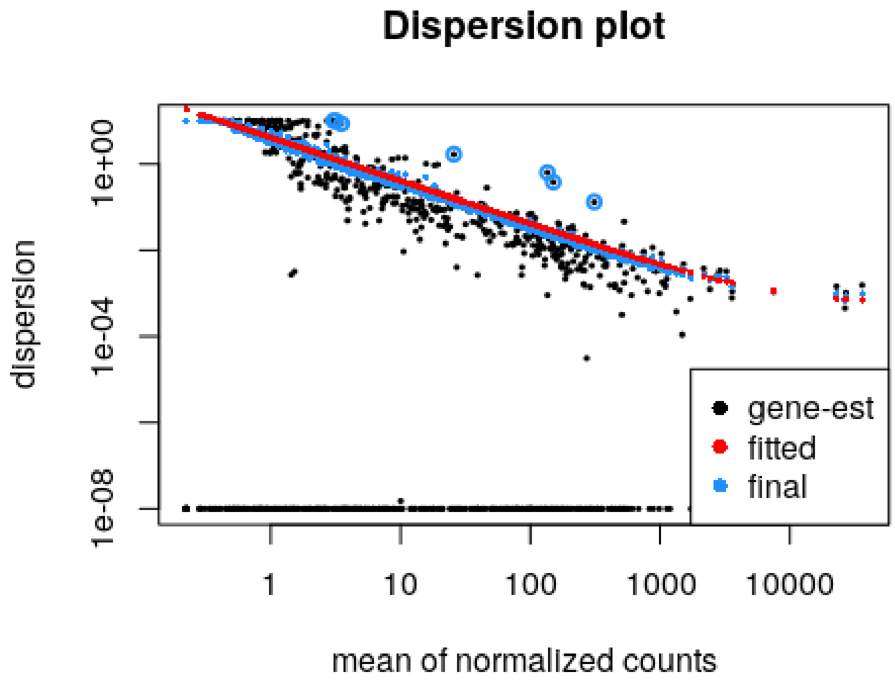


```
pcaData <- plotPCA (rld, intgroup=c("group", "con"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=group, shape=con)) +
        geom_point(size=3) +
        xlab(paste0("PC1: ",percentVar[1],"% variance")) +
        ylab(paste0("PC2: ",percentVar[2],"% variance")) +
        coord_fixed()
```



DESeq uses a negative binomial distribution.   the negative binomial distribution is a discrete probability distribution of the number of successes in a sequence of independent and identically distributed Bernoulli trials before a specified (non-random) number of failures (denoted r) occurs. Two parameters: mean and dispersion.

```
plotDispEsts(dds_con, main="Dispersion plot"
```



It is time to look at some p-values:

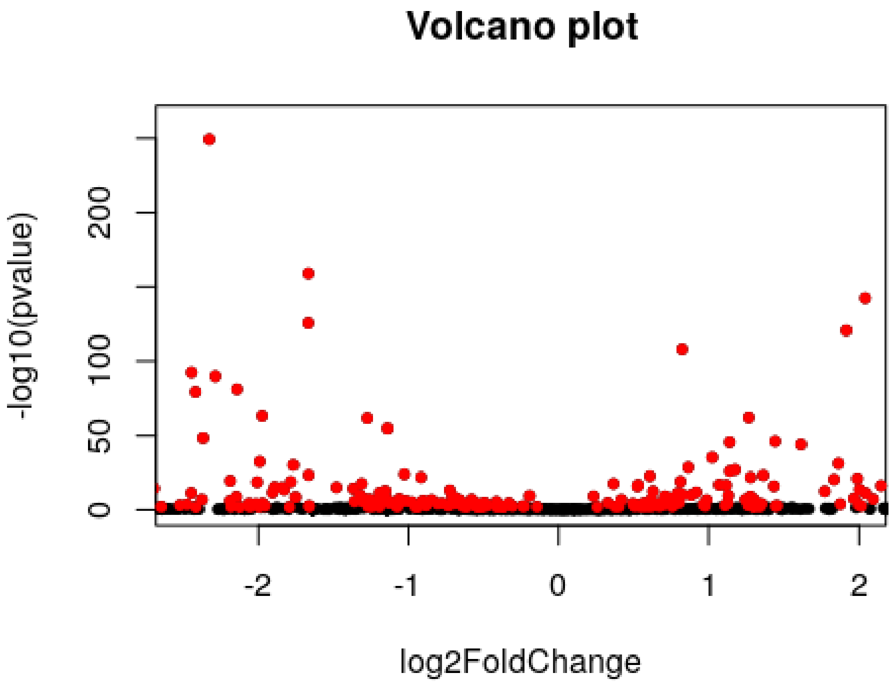
```
table(res_con$padj<0.05)

##
## FALSE  TRUE
##   403   339

res_con <- res_con[order(res_con$padj), ]
```

Volcano Plot

```
with(res_con, plot(log2FoldChange, -log10(pvalue), pch=20, main="Volcano
plot", xlim=c(-2.5,2)))
    with(subset(res_con, padj<.05 ), points(log2FoldChange, -log10(pvalue),
pch=20, col="red"))
```

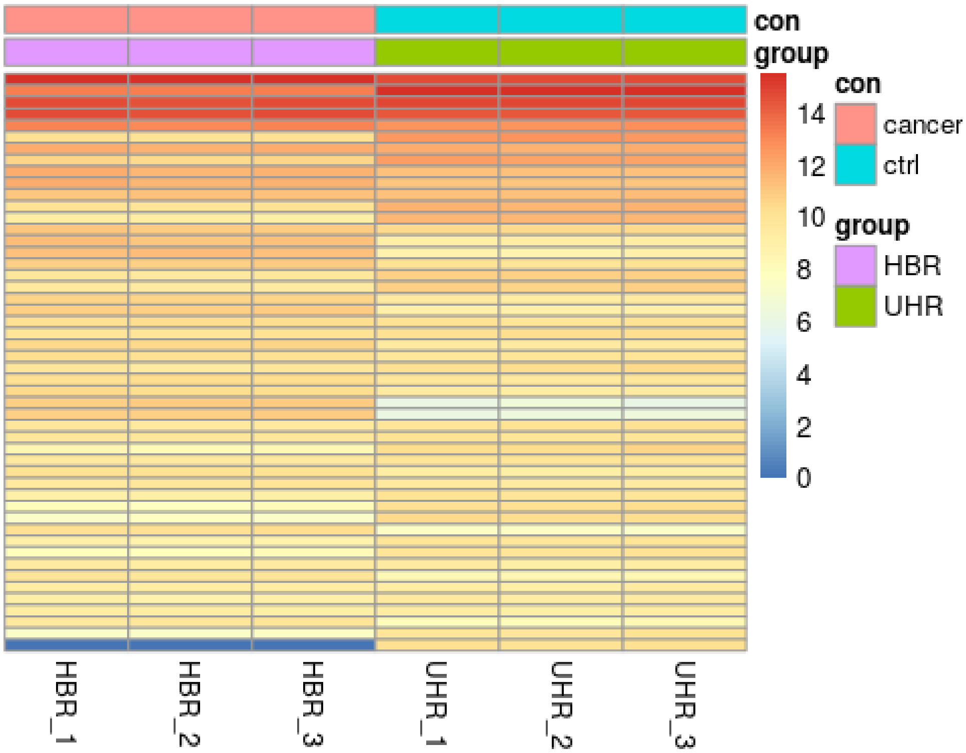


Heatmap of count matrix Version 1:

Heatmap of normalized dds counts

```
library("pheatmap")
select <- order(rowMeans(counts(dds,normalized=TRUE)),decreasing=TRUE) [1:50]
nt <- normTransform(dds) # defaults to log2(x+1)
log2.norm.counts <- assay(nt)[select,]
df <- as.data.frame(colData(dds)[,c("group","con")])

pheatmap(log2.norm.counts, cluster_rows=FALSE, show_rownames=FALSE,
cluster_cols=FALSE, annotation_col=df, fontsize_row = 5)
```

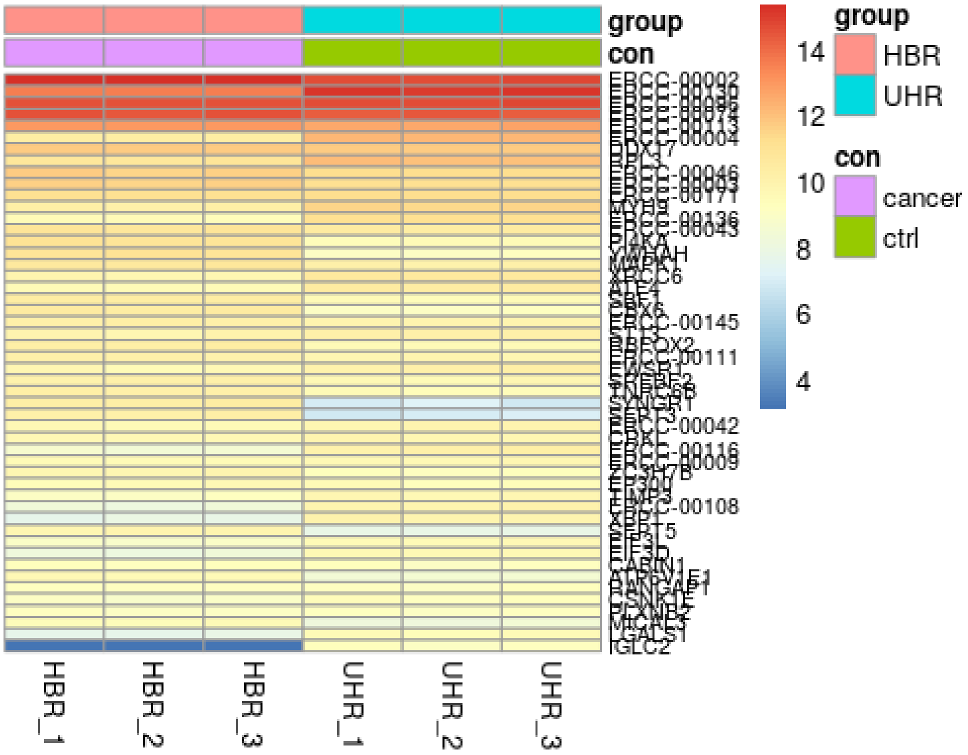


Version2:

Heatmap of regularized log transformed dds counts

```
df <- as.data.frame(colData(rld)[,c("con","group")])


pheatmap(assay(rld)[select,], cluster_rows=FALSE, show_rownames=TRUE,
cluster_cols=FALSE, annotation_col=df, fontsize_row =8)
```



Merge results with normalized count data

```
resdata <- merge(as.data.frame(res_con), as.data.frame(counts(dds_con,
normalized=TRUE)), by="row.names", sort=FALSE)
names(resdata)[1] <- "Gene"
head(resdata)
```

```
##          Gene   baseMean log2FoldChange      lfcSE      stat        pvalue
## 1      SYNGR1   986.7215       4.662950 0.11914480  39.13683  0.000000e+00
## 2       SEPT3   926.8501       4.527019 0.12032694  37.62266  0.000000e+00
## 3 ERCC-00004  3589.9516      -2.501269 0.05482358 -45.62396  0.000000e+00
## 4 ERCC-00130 27062.3042      -2.175087 0.03596987 -60.46970  0.000000e+00
## 5       YWHAH  1474.2754       2.530765 0.07315383  34.59512 2.991523e-262
## 6 ERCC-00136  1727.3486      -2.326855 0.06889317 -33.77483 4.619251e-250
##              padj     HBR_1      HBR_2     HBR_3       UHR_1       UHR_2
## 1  0.000000e+00 1852.7882  1902.3180 1937.5307    71.74826    94.84367
## 2  0.000000e+00 1710.3648  1728.1621 1890.0422    67.72500    74.44718
## 3  0.000000e+00 1098.3288  1032.5691 1104.1076  5942.36562  6377.98212
## 4  0.000000e+00 9713.0242 10113.4068 9606.9230 44973.42155 42345.15087
## 5 4.439420e-260 2478.9383  2572.1483 2490.7717   431.16013   407.92978
## 6 5.712474e-248  603.0543   585.3286  534.2456  2865.90732  2884.06355
##          UHR_3
## 1     61.09999
## 2     90.35914
## 3   5984.35667
## 4  45621.89887
## 5    464.70414
## 6   2891.49244
```

```
write.table (resdata, file = “/home/mlsi/RNASeq/analysis/DESeq2/diffexpr-results_RNASeq.txt”, sep = " ", col.names=NA)

resdata_GSEA<- resdata[ ,-(2:7)] write.table (resdata_GSEA, file = “/home/mlsi/RNASeq/analysis/DESeq2/diffexpr-results_RNASeq_GSEA.txt”, sep = ", col.names=NA)

resdata_GSEA<- read.table (“/home/mlsi/RNASeq/analysis/DESeq2/diffexpr-results_RNASeq_GSEA.txt”, header= TRUE, row.names=2) resdata_GSEA<- resdata_GSEA_2 [ ,-1] write.table (resdata_GSEA, file = “/home/mlsi/RNASeq/analysis/DESeq2/diffexpr-results_RNASeq_GSEA.txt”, sep = ", col.names=NA)
```