BIOMED. R TUSHAL KAKADIYA

Loading FeatureCount Table

First of we load Load FeatureCountTable into R to begin our Analysis.

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

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

Edit FeatureCountTable

Removing first five columns (chr, start, end, strand, length, etc.)

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

Removeing .bam or -.sam from filenames

```
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"
```

Converting the data.frame into a matrix

The next step is use to convert the data.frame into a matrix because DESeq2 needs matrix.

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

## [1] "matrix"
```

here “matrix” is a class of our countdata folder. ## Design coldata Here we are grouping columns of our sample data of UHR, HBR. because here we are performing task with normal vs cancer data.

```
group<- factor(c(rep("UHR",3), rep("HBR",3)))
con<- factor(c(rep("Normal",3), rep("Cancer",3)))
```

Create coldata frame

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

colors for plots

Here we are defining the colours for Upcoming ploat in our case we choose Paired colour plot from ColorBrewer.

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

Create DESeqDataSet

Here we Instantiating DESeqDataset but first we have to install library called DESeq2

```
library(DESeq2)

## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## 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, cbind, colMeans, colnames,
##     colSums, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, lengths, Map, mapply, match,
##     mget, order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## 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")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

##
## Attaching package: 'DelayedArray'

## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges

## The following object is masked from 'package:base':
##
##     apply

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

To see an overview of counts we perform next command

```
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
```

Create the estimateSizeFactors. To create data table with read counts normalized to library size,The following commands are usefull:(possible before or after Pre-filtering)

```
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

Used to Filter out lowly expressed genes and check the dimension of the dataset:

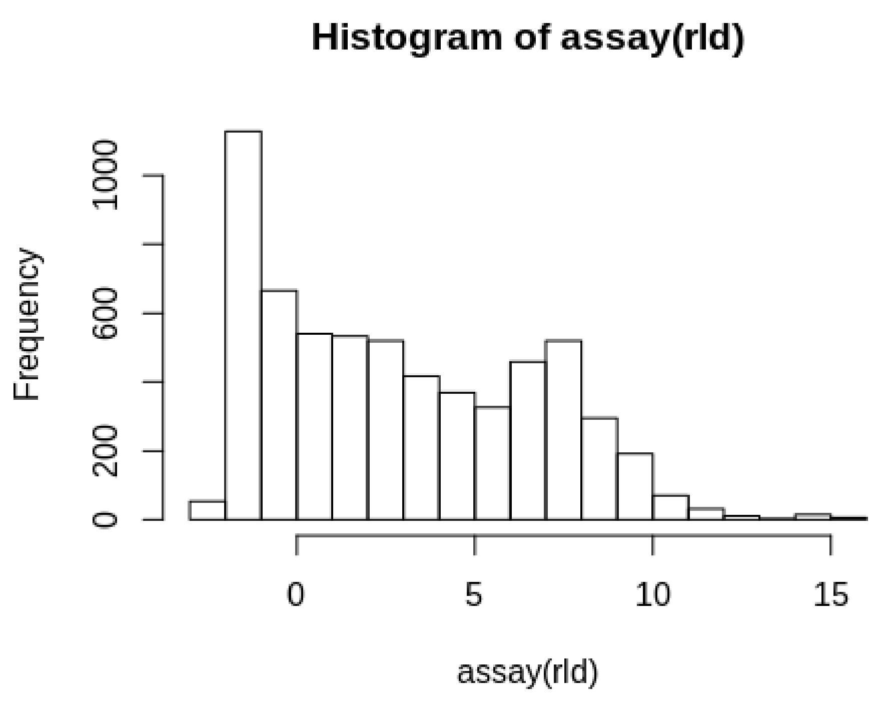
```
dds<- dds [rowSums(counts(dds)) > 0, ]
dim(dds)
```

```
## [1] 1026    6
```

Rlog Transformation

It is Use the Regularized log transformation for clustering/heatmaps, etc of Count Data.

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

##                    HBR_1      HBR_2      HBR_3      UHR_1      UHR_2      UHR_3
## LA16c-60D12.1 -0.416971 -0.4388245 -0.4256907  0.6260911  0.6163041 -0.4555020
## LA16c-13E4.3  -1.983069 -1.9949827 -1.9877230 -2.0192063 -1.9958550 -1.8952429
## LA16c-60D12.2 -0.487308 -0.5147786 -0.4982003 -0.5655179  1.2011416  0.1467634
## ZNF72P        -1.543169 -1.5674099 -1.5526386 -1.6166974 -1.1235832 -1.5872711
## BNIP3P2       -1.543169 -1.5674099 -1.5526386 -1.6166974 -1.1235832 -1.5872711
## LA16c-60G3.6  -1.543169 -1.5674099 -1.5526386 -1.6166974 -1.1235832 -1.5872711

hist(assay(rld))
```



Now we are performing Differential Expression Analysis via DESeq2. It clearly tells results which comparison to make by setting factor levels:

```
dds$con<- relevel(dds$con, ref="Cancer")
```

Now check the design of the current DESeqDataSet(dds)

```
design(dds)

## ~con
```

Here we 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
```

Now we are check factor setting and find possible comparisons.

```
resultsNames(dds_con)

## [1] "Intercept"            "con_Normal_vs_Cancer"
```

To Create results tables we use following command

```
res_con<- results(dds_con, contrast=c("con", "Normal", "Cancer"))
```

Here we are Exploring the result tables

```
summary(res_con)

##
## out of 1026 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)     : 180, 18%
## LFC < 0 (down)    : 200, 19%
## outliers [1]      : 0, 0%
## low counts [2]    : 239, 23%
## (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 Normal vs Cancer
## lfcSE               results         standard error: con Normal vs Cancer
## stat                results         Wald statistic: con Normal vs Cancer
## pvalue              results      Wald test p-value: con Normal vs Cancer
## padj                results                     BH adjusted p-values
```

Here we are Changing 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

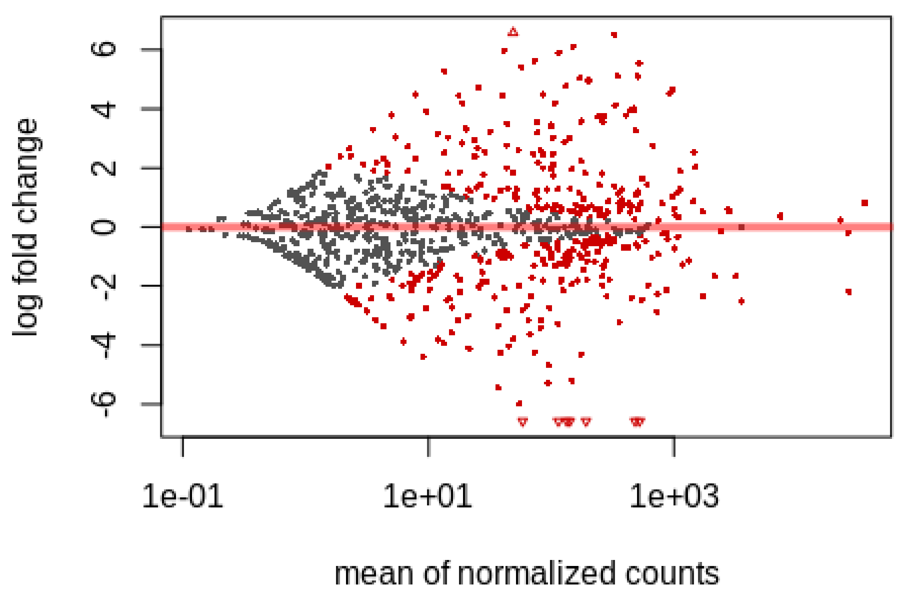
Shrinkage Shrinkage of effect size (LFC estimates) is useful for visualization and ranking of genes.

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

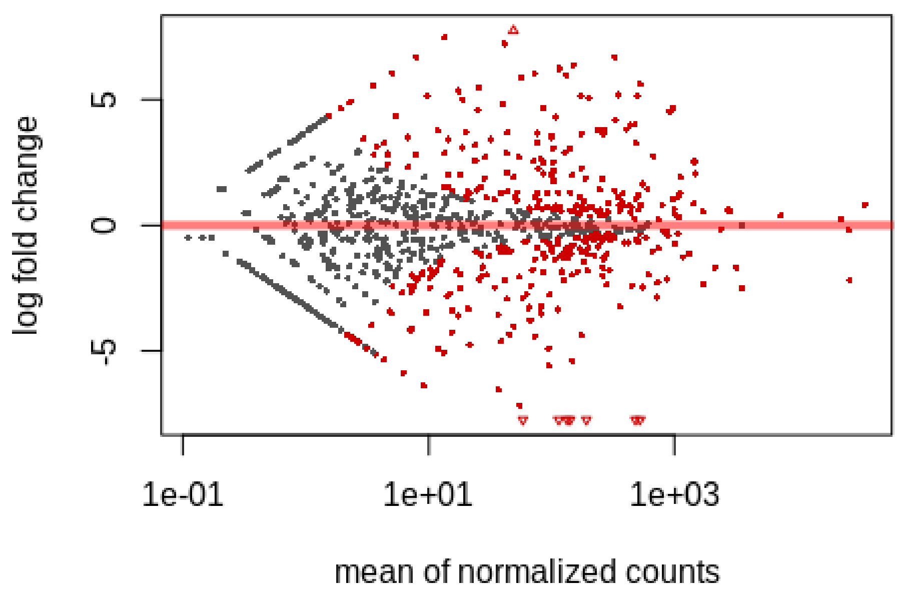
plotMA

plotMA In DESeq2, 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.

```
plotMA(resLFC_con)
```

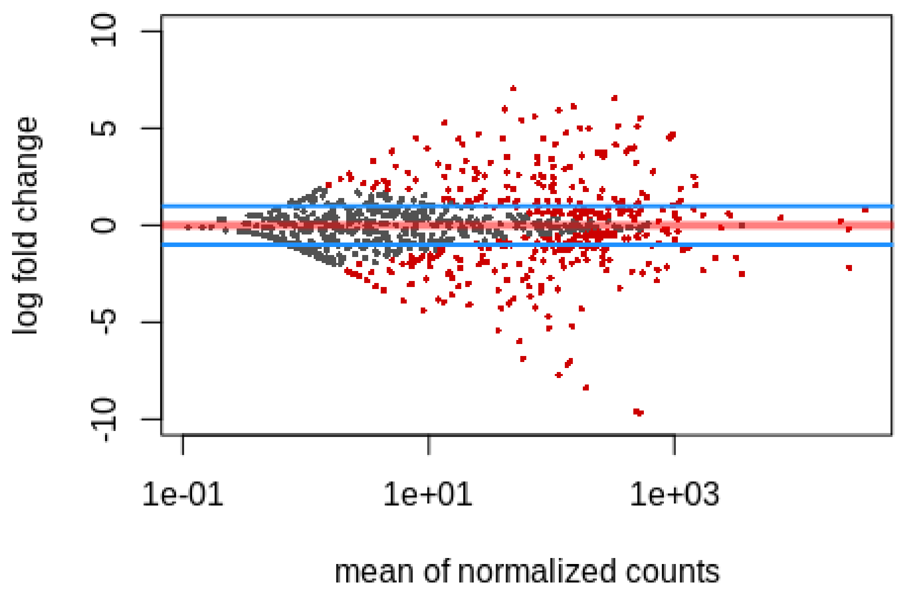


```
plotMA(res_con)
```

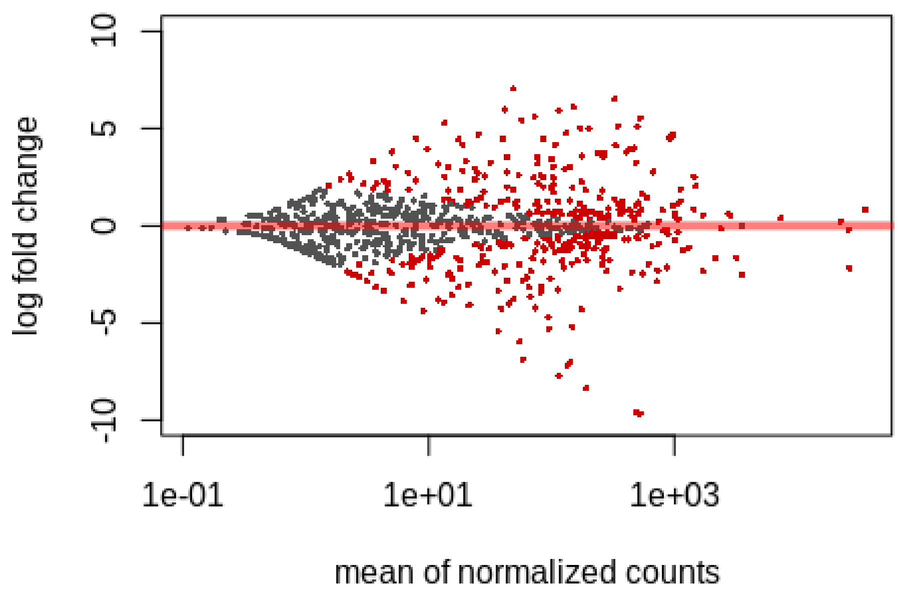


Abline: h : the y-value(s) for horizontal line(s) [v : the x-value(s) for vertical line(s)] a, b : single values specifying the intercept and the slope of the line.

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



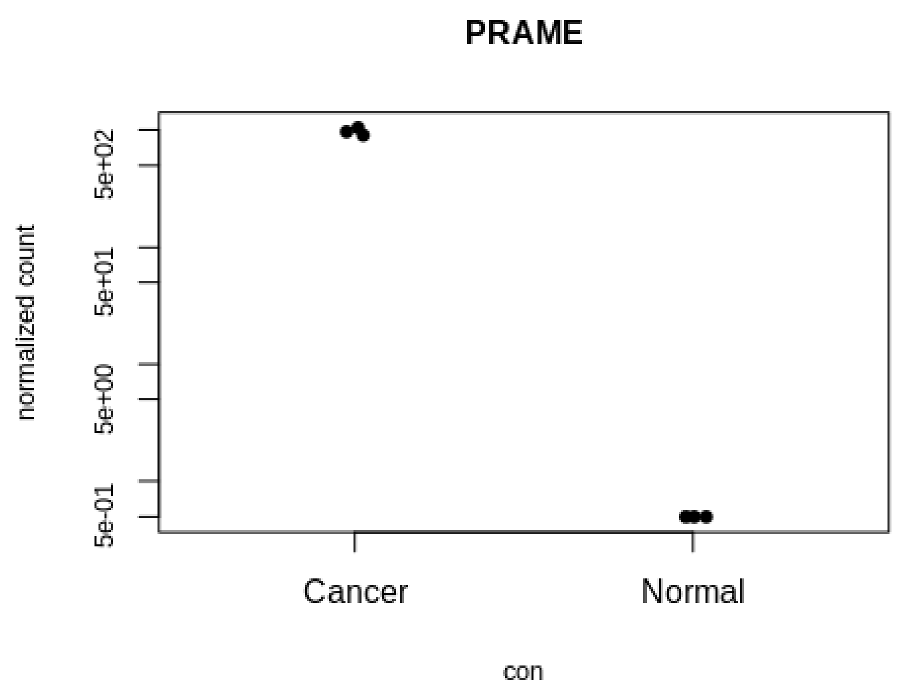
```
plotMA(resLFC_con, ylim=c(-10,10))
idx <- identify(resLFC_con$baseMean, resLFC_con$log2FoldChange)
```



```
library("ggplot2")
```

Visualizing counts of a single gene of interest via plotCounts:

```
plotCounts(dds_con,gene="PRAME", 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 heatmape

Sample distance heatmap A heatmap of this distance matrix gives us an overview over similarities and dissimilarities between samples. Multiple versions are possible.

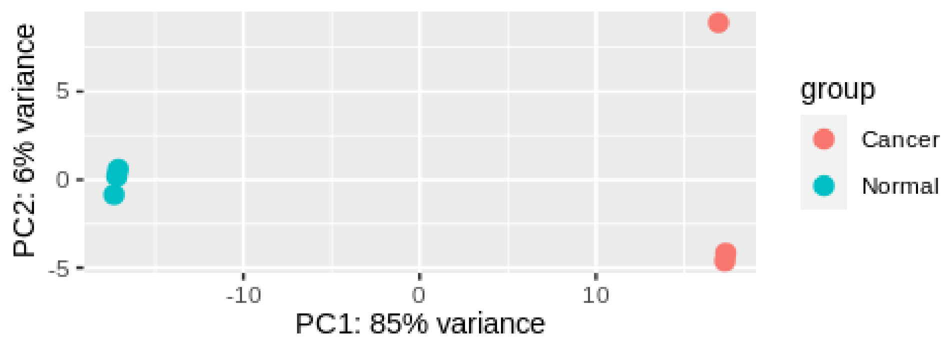
Example 1: Convert regulized log transferred count data into a sample-dist-matrix:

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

Principal Component plot

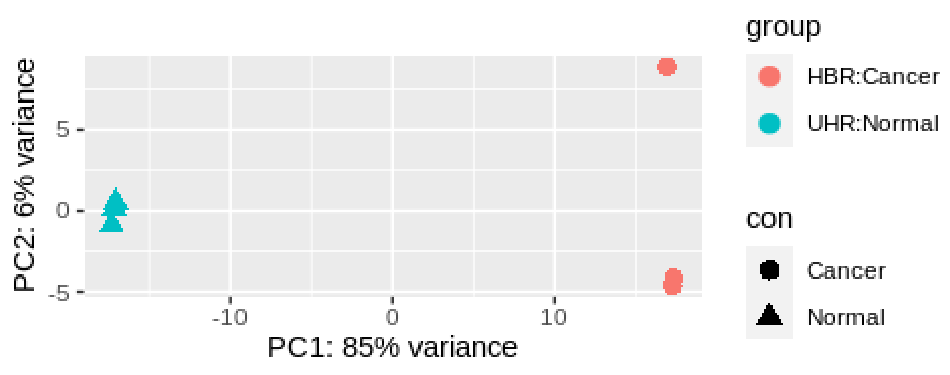
Related to the distance matrix is the PCA plot, which shows the samples in the 2D plane spanned by their first two principal components. This type of plot is useful for visualizing the overall effect of experimental covariates and batch effects.

```
plotPCA(rld, intgroup="con")
```



To distinguish between two different conditions within on PCA plot

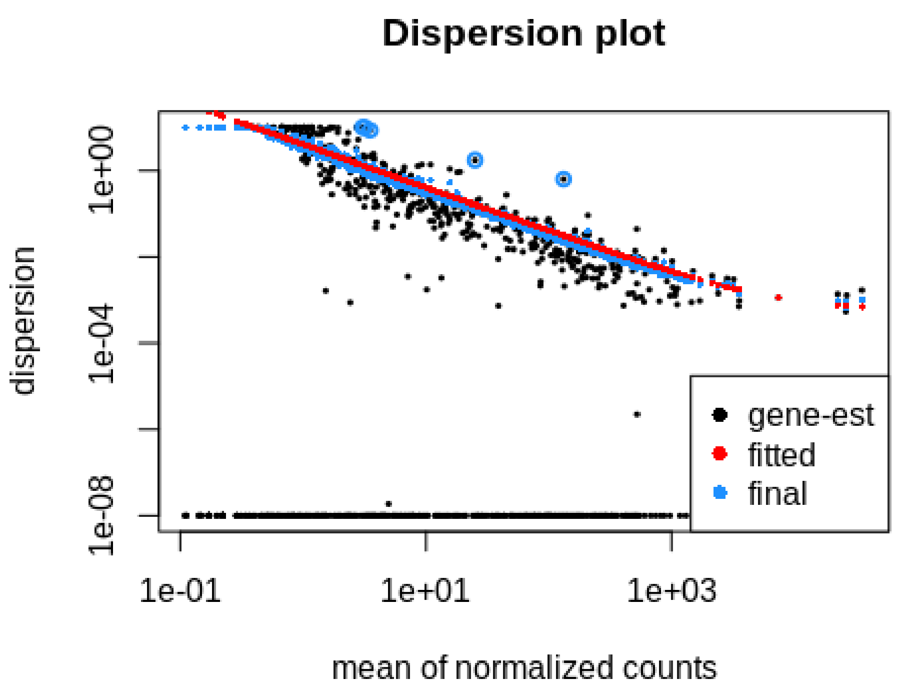
```
pcaData <- plotPCA (rld, intgroup=c("group", "con"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=group, shape=con)) +
geom_point(size=3) +
xlab(paste0("PC1: ",percentVar[1],"% variance")) +
ylab(paste0("PC2: ",percentVar[2],"% variance"))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



## Plot dispersions

DESeq uses a 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")
```



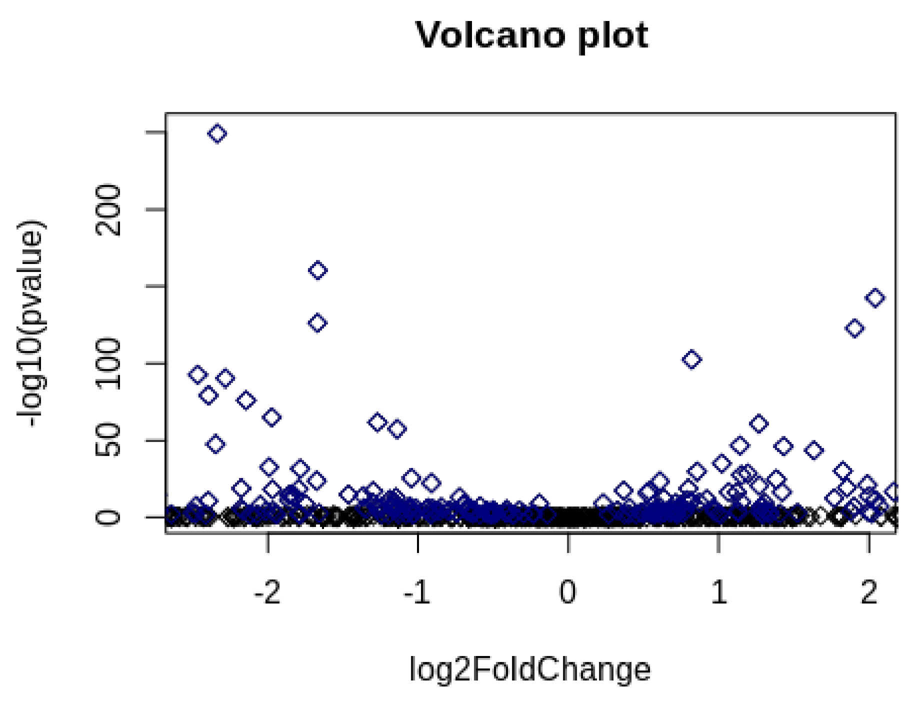
This command will show us a P-value of our Data.

```
table(res_con$padj<0.05)

## 
## FALSE  TRUE 
##   447   340

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

## log2 fold change (MLE): con Normal vs Cancer 
## Wald test p-value: con Normal vs Cancer 
## DataFrame with 1026 rows and 6 columns
##               baseMean log2FoldChange     lfcSE      stat       pvalue
##              <numeric>      <numeric> <numeric> <numeric>    <numeric>
## SYNGR1        972.8030       4.664819 0.11866380  39.31122 0.000000e+00
## SEPT3         918.0706       4.535216 0.11990296  37.82405 0.000000e+00
## ERCC-00004   3548.4375      -2.504053 0.05415789 -46.23617 0.000000e+00
## ERCC-00130  26696.3423      -2.177783 0.03744874 -58.15370 0.000000e+00
## YWHAH       1446.7985       2.524186 0.07433991  33.95466 1.041182e-252
## ...                ...            ...       ...       ...          ...
## ERCC-00120   0.4293669      2.4599421  4.027175 0.6108357    0.5413084
## ERCC-00134   0.3943751      2.3527178  4.036308 0.5828885    0.5599683
## ERCC-00138   0.7693084      0.3617995  3.117796 0.1160434    0.9076182
## ERCC-00142   0.3399415     -1.6336205  4.018451 -0.4065299   0.6843533
## ERCC-00164   0.3454011      2.1872220  4.051758 0.5398205    0.5893208
##                     padj
##                <numeric>
## SYNGR1      0.000000e+00
## SEPT3       0.000000e+00
## ERCC-00004  0.000000e+00
## ERCC-00130  0.000000e+00
## YWHAH      1.638821e-250
## ...                  ...
## ERCC-00120            NA
## ERCC-00134            NA
## ERCC-00138            NA
## ERCC-00142            NA
## ERCC-00164            NA
```

Volcano Plot

A volcano plot is a type of scatterplot that shows statistical significance (P value) versus magnitude of change (fold change). It enables quick visual identification of genes with large fold changes that are also statistically significant. These may be the most biologically significant genes.

```
with(res_con, plot(log2FoldChange, -log10(pvalue), pch=23, main="Volcano plot", xlim=c(-2.5,2)))
with(subset(res_con, padj<.05 ), points(log2FoldChange, -log10(pvalue), pch=23,
col="navyblue"))
```



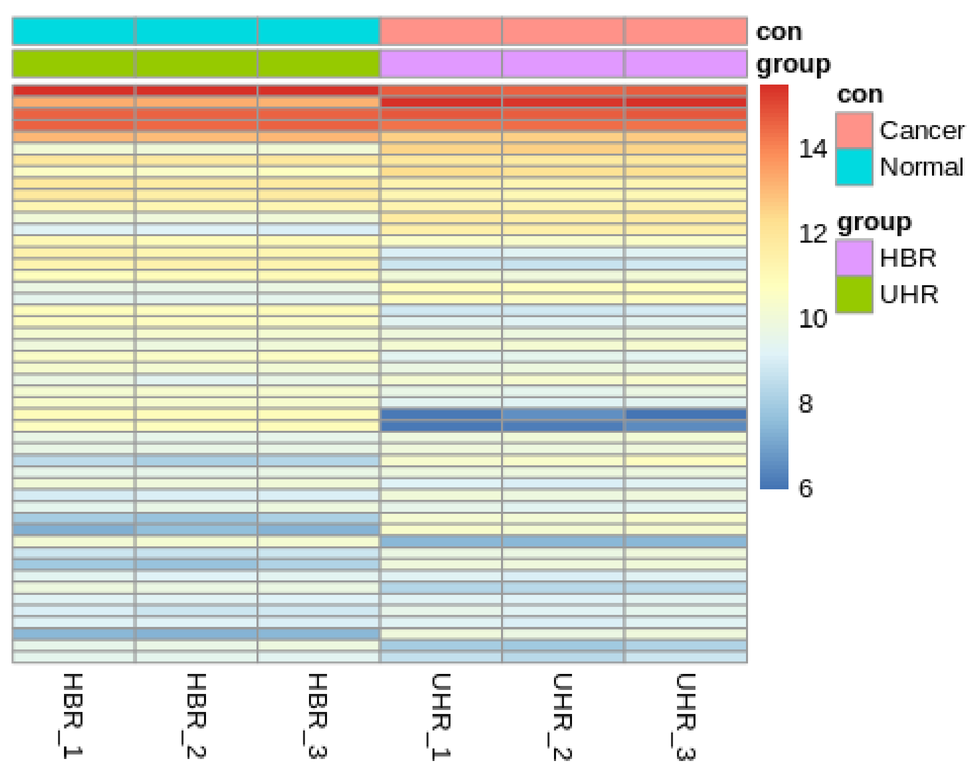
## Heatmap of count matrix

```
install.packages("pheatmap")

## Installing package into '/home/mlsi/R_libs'
## (as 'lib' is unspecified)
```

Version 1 of Heat map

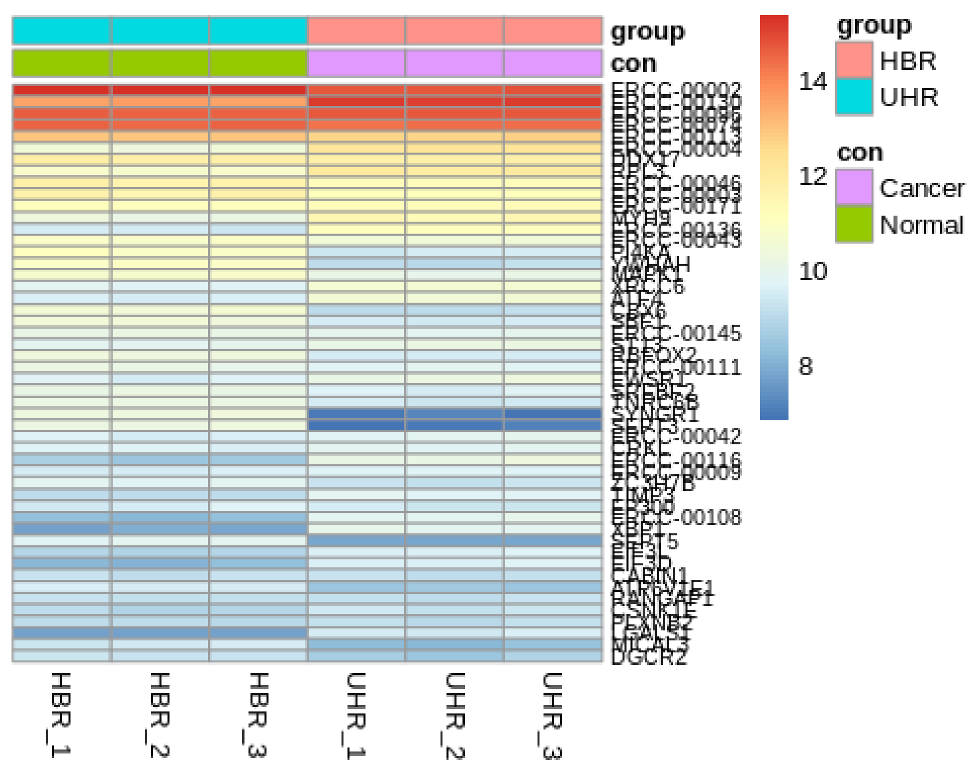
The below heat map is for normalized dds counts

```
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)
```



### Version-2 The below Heatmap of regularized log is for the transformed dds counts as we can see right an side

```
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 In Next steps we are merging the results about heatmap 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   972.8030       4.664819 0.11866380  39.31122  0.000000e+00
## 2      SEPT3   918.0706       4.535216 0.11990296  37.82405  0.000000e+00
## 3 ERCC-00004  3548.4375      -2.504053 0.05415789 -46.23617  0.000000e+00
## 4 ERCC-00130 26696.3423      -2.177783 0.03744874 -58.15370  0.000000e+00
## 5      YWHAH  1446.7985       2.524186 0.07433991  33.95466 1.041182e-252
## 6 ERCC-00136  1706.4092      -2.337733 0.06923717 -33.76414 6.631072e-250
##            padj     HBR_1     HBR_2     HBR_3       UHR_1       UHR_2
## 1  0.000000e+00 1838.1198 1867.2382 1907.1979    69.90157    92.80403
## 2  0.000000e+00 1688.7001 1721.1335 1869.3379    67.23865    74.44718
## 3  0.000000e+00 1084.5808 1023.7688 1083.7427  5862.41136  6290.27722
## 4  0.000000e+00 9580.8934 9999.3608 9410.5783 44231.04820 41690.42357
## 5 1.638821e-250 2437.0866 2546.9874 2412.3924   426.06669   398.75136
## 6 8.697756e-248  589.9501  580.2738  520.5751  2825.35473  2847.34987
##         UHR_3
## 1    61.55656
## 2    87.56638
## 3  5945.84392
## 4 45265.74965
## 5   459.50675
## 6  2874.95165
```

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

```
resdata_GSEA<- resdata[ ,-(2:7)]
write.table (resdata_GSEA, file = "/home/mlsi/RNASeq/analysis/DESeq2/diffexprresults_
RNASeq_GSEA.txt", sep = "\t", col.names=NA)

resdata_GSEA<- read.table ("/home/mlsi/RNASeq/analysis/DESeq2/diffexprresults_
RNASeq_GSEA.txt", header= TRUE, row.names=2)
resdata_GSEA<- resdata_GSEA [ ,-1]
write.table (resdata_GSEA, file = "/home/mlsi/RNASeq/analysis/DESeq2/diffexprresults_
RNASeq_GSEA.txt", sep = "\t ", col.names=NA)
```