Differential Expression Analysis Using Deseq2

Monica Mbabazi & Eric Kariuki - Star Output

8/08/2020

Creating the DESeq2 object and annotating the metadata

This step uses the counts output generated from *STAR* alignment in Bash.

```
knitr::opts_chunk$set(echo = TRUE)

#library(tidyverse)
library(BiocManager)
```

```
## Bioconductor version 3.11 (BiocManager 1.30.10), ?BiocManager::install for help
```

```
#BiocManager::install('DESeq2')
library(kableExtra)#creating decent tables for HTML and PDF
options(knitr.table.format = "latex")
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, 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, which.max, which.min

##
## Attaching package: 'S4Vectors'

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

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## 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 objects are masked from 'package:base':
##
##     aperm, apply, rowsum
```

```
library(Rsubread)
staroutput <- read.table(file = 'C:/Users/SecondFiddle/Desktop/qcrepo/RNAseq_miniproj/STAR_counts.txt',
#Remove unwanted columns
staroutput <- staroutput[, -c(1:5)]
head(staroutput)
```

```
##                 STAR_Alignment.sample37_sorted.bam
## ENSG00000223972                                  0
## ENSG00000227232                                 96
## ENSG00000278267                                  8
## ENSG00000243485                                  0
## ENSG00000284332                                  0
## ENSG00000237613                                  0
##                 STAR_Alignment.sample38_sorted.bam
## ENSG00000223972                                  0
## ENSG00000227232                                 72
## ENSG00000278267                                 21
## ENSG00000243485                                  0
## ENSG00000284332                                  0
## ENSG00000237613                                  0
##                 STAR_Alignment.sample39_sorted.bam
## ENSG00000223972                                  0
## ENSG00000227232                                265
## ENSG00000278267                                 79
## ENSG00000243485                                  0
## ENSG00000284332                                  0
## ENSG00000237613                                  0
##                 STAR_Alignment.sample40_sorted.bam
## ENSG00000223972                                  0
## ENSG00000227232                                 59
## ENSG00000278267                                  1
## ENSG00000243485                                  0
## ENSG00000284332                                  0
## ENSG00000237613                                  0
##                 STAR_Alignment.sample41_sorted.bam
## ENSG00000223972                                  0
## ENSG00000227232                                 62
## ENSG00000278267                                  0
## ENSG00000243485                                  0
## ENSG00000284332                                  0
## ENSG00000237613                                  0
##                 STAR_Alignment.sample42_sorted.bam
```

```
## ENSG00000223972                                 0
## ENSG00000227232                                61
## ENSG00000278267                                 6
## ENSG00000243485                                 0
## ENSG00000284332                                 0
## ENSG00000237613                                 0
```

```
staroutput <- as.matrix(staroutput)

#Reading in the metadata
metadata <- read.table(file = 'C:/Users/SecondFiddle/Desktop/qcrepo/RNAseq_miniproj/metadata.tsv', sep =
head(staroutput)
```

```
##                 STAR_Alignment.sample37_sorted.bam
## ENSG00000223972                                 0
## ENSG00000227232                                96
## ENSG00000278267                                 8
## ENSG00000243485                                 0
## ENSG00000284332                                 0
## ENSG00000237613                                 0
##                 STAR_Alignment.sample38_sorted.bam
## ENSG00000223972                                 0
## ENSG00000227232                                72
## ENSG00000278267                                21
## ENSG00000243485                                 0
## ENSG00000284332                                 0
## ENSG00000237613                                 0
##                 STAR_Alignment.sample39_sorted.bam
## ENSG00000223972                                 0
## ENSG00000227232                               265
## ENSG00000278267                                79
## ENSG00000243485                                 0
## ENSG00000284332                                 0
## ENSG00000237613                                 0
##                 STAR_Alignment.sample40_sorted.bam
## ENSG00000223972                                 0
## ENSG00000227232                                59
## ENSG00000278267                                 1
## ENSG00000243485                                 0
## ENSG00000284332                                 0
## ENSG00000237613                                 0
##                 STAR_Alignment.sample41_sorted.bam
## ENSG00000223972                                 0
## ENSG00000227232                                62
## ENSG00000278267                                 0
## ENSG00000243485                                 0
## ENSG00000284332                                 0
## ENSG00000237613                                 0
##                 STAR_Alignment.sample42_sorted.bam
## ENSG00000223972                                 0
## ENSG00000227232                                61
## ENSG00000278267                                 6
## ENSG00000243485                                 0
## ENSG00000284332                                 0
```

```
## ENSG00000237613                                0
```

```
#View(metadata)

colnames(staroutput) <-metadata[, 1]

#Reading the data into Deseq
Condition <- factor(c(rep("normal", 3), rep("disease", 3)))
col.data <- data.frame(row.names = colnames(staroutput), Condition)
#colnames(staroutput) <- gsub("STAR_Alignment.", "", colnames(staroutput), fixed = TRUE)
#colnames(staroutput) <- gsub("_sorted.bam", "", colnames(staroutput), fixed = TRUE)
#staroutput
head(col.data)
```

```
##          Condition
## sample37    normal
## sample38    normal
## sample39    normal
## sample40   disease
## sample41   disease
## sample42   disease
```

```
StarDeseq <-DESeqDataSetFromMatrix(countData = staroutput,
                                   colData = col.data,
                                   design = ~ Condition)
#StarDeseq<- log2(staroutput + 1) # log-transform to make numbers on scale (+1 to avoid zeroes)
#boxplot(StarDeseq)
#estimateSizeFactors(StarDeseq, type = 'iterate')
StarDeseq
```

```
## class: DESeqDataSet
## dim: 60683 6
## metadata(1): version
## assays(1): counts
## rownames(60683): ENSG00000223972 ENSG00000227232 ... ENSG00000277475
##   ENSG00000268674
## rowData names(0):
## colnames(6): sample37 sample38 ... sample41 sample42
## colData names(1): Condition
```

```
StarDeseq <- DESeq(StarDeseq)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates
```

```
## fitting model and testing
```

```
#res <- results(A)
#write.table(x = staroutput, 'C:/Users/SecondFiddle/Desktop/qcrepo/RNAseq_miniproj/staroutput')

res <- results(StarDeseq)

head(res, tidy=TRUE) #A glimpse at the results table
```

```
## log2 fold change (MLE): Condition normal vs disease
## Wald test p-value: Condition normal vs disease
## DataFrame with 6 rows and 6 columns
##                  baseMean log2FoldChange     lfcSE      stat     pvalue
##                 <numeric>      <numeric> <numeric> <numeric>  <numeric>
## ENSG00000223972    0.0000             NA        NA        NA         NA
## ENSG00000227232   96.2073        1.25705   0.52688   2.38583 0.01704042
## ENSG00000278267   15.9563        3.92119   1.27383   3.07827 0.00208207
## ENSG00000243485    0.0000             NA        NA        NA         NA
## ENSG00000284332    0.0000             NA        NA        NA         NA
## ENSG00000237613    0.0000             NA        NA        NA         NA
##                      padj
##                 <numeric>
## ENSG00000223972        NA
## ENSG00000227232 0.1393479
## ENSG00000278267 0.0378556
## ENSG00000243485        NA
## ENSG00000284332        NA
## ENSG00000237613        NA
```

```
summary(res) #summary of the results
```

```
##
## out of 37291 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 956, 2.6%
## LFC < 0 (down)     : 1447, 3.9%
## outliers [1]       : 289, 0.77%
## low counts [2]     : 12584, 34%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
#Sorting summary list by pvalue
res <- res[order(res$padj),]
head(res)
```

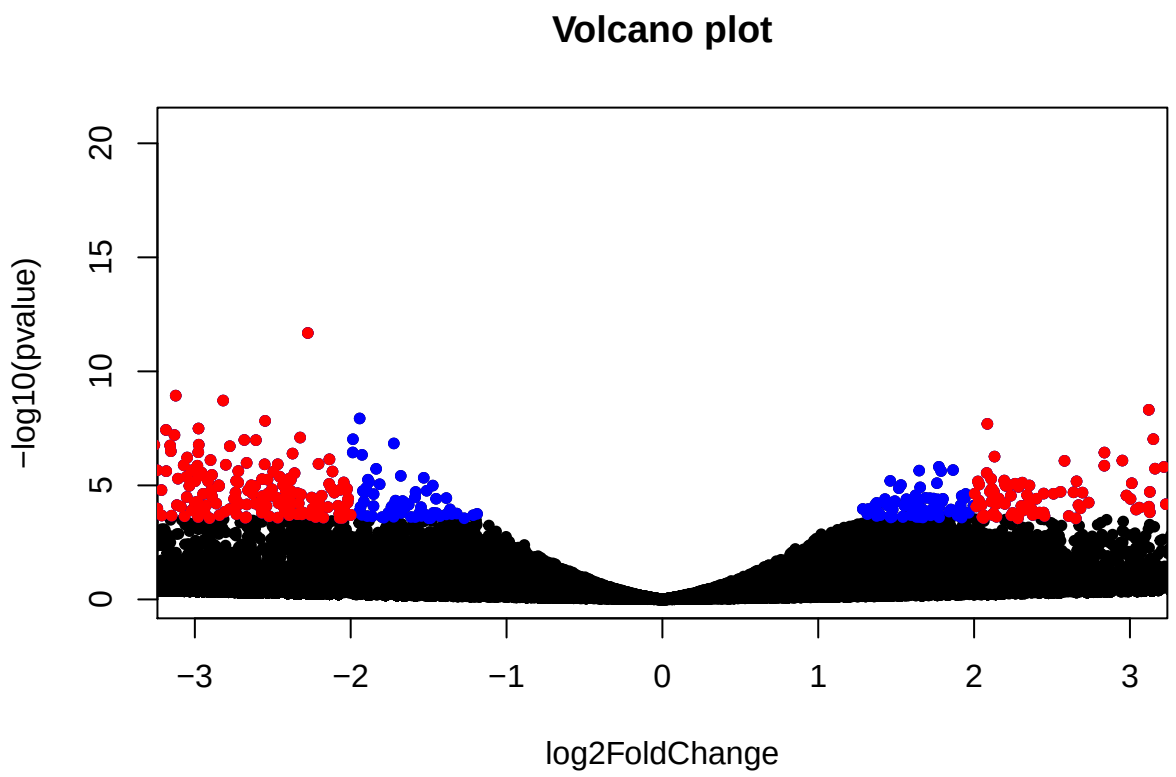
```
## log2 fold change (MLE): Condition normal vs disease
## Wald test p-value: Condition normal vs disease
## DataFrame with 6 rows and 6 columns
##                  baseMean log2FoldChange     lfcSE      stat      pvalue
##                 <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000039537   432.692       -6.92906  0.728306  -9.51394 1.83584e-21
```

```
## ENSG00000124237   241.987       -7.99243  0.856306  -9.33361 1.02322e-20
## ENSG00000160401   508.113       -6.04049  0.657253  -9.19051 3.90994e-20
## ENSG00000007908  1329.338       -6.21218  0.690436  -8.99748 2.30961e-19
## ENSG00000188817   409.416       -5.83075  0.665742  -8.75827 1.98275e-18
## ENSG00000168658   330.183      -11.73876  1.396799  -8.40404 4.31376e-17
##                       padj
##                  <numeric>
## ENSG00000039537 4.48274e-17
## ENSG00000124237 1.24925e-16
## ENSG00000160401 3.18243e-16
## ENSG00000007908 1.40990e-15
## ENSG00000188817 9.68296e-15
## ENSG00000168658 1.65997e-13
```

```r
#Volcano plot
par(mfrow=c(1,1))
# Make a basic volcano plot

with(res, plot(log2FoldChange, -log10(pvalue), pch=20, main="Volcano plot", xlim=c(-3,3)))
#For the adjusted p value
# Add colored points: blue if padj<0.01, red if log2FC>1 and padj<0.05)

with(subset(res, padj<.01 ), points(log2FoldChange, -log10(pvalue), pch=20, col="blue"))
with(subset(res, padj<.01 & abs(log2FoldChange)>2), points(log2FoldChange, -log10(pvalue), pch=20, col=
```



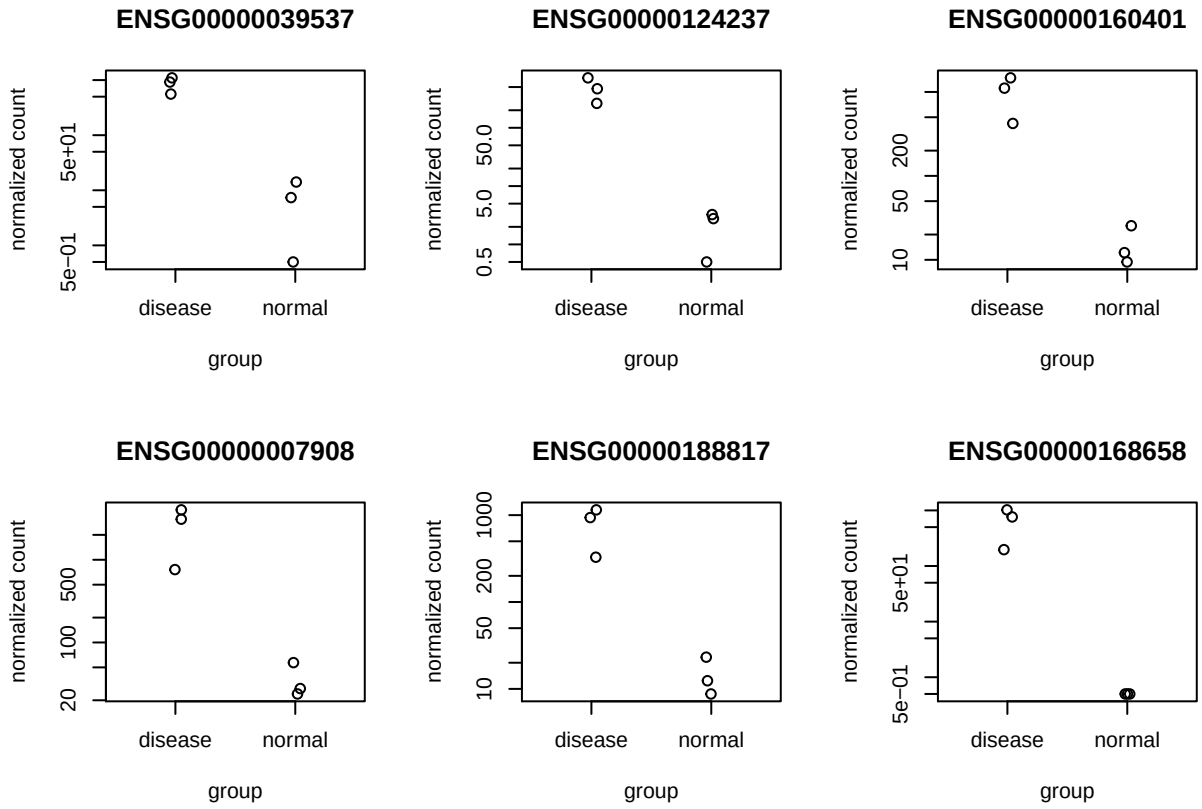
```r
#Changing the colors of the plots
library(RColorBrewer)
(mycols <- brewer.pal(8, "Dark2")[1:length(unique(Condition))])
```

```
## [1] "#1B9E77" "#D95F02"
```

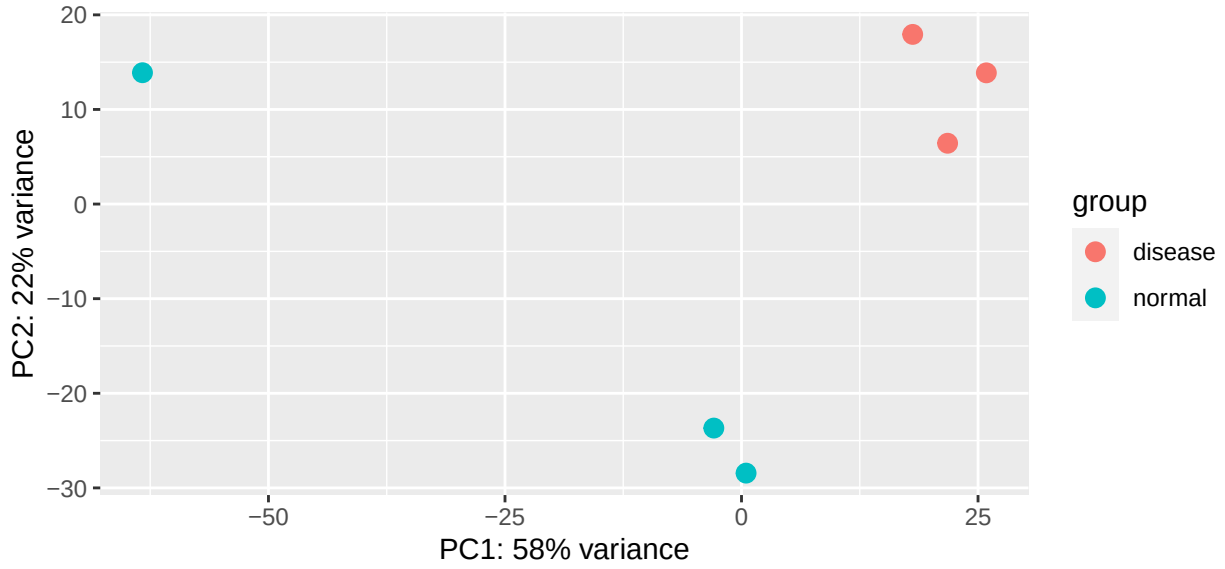
```r
#Using the PlotCounts function to compare the normalized counts between treated and untreated groups

par(mfrow=c(2,3))

plotCounts(StarDeseq, gene="ENSG00000039537", intgroup="Condition")
plotCounts(StarDeseq, gene="ENSG00000124237", intgroup="Condition")
plotCounts(StarDeseq, gene="ENSG00000160401", intgroup="Condition")
plotCounts(StarDeseq, gene="ENSG00000007908", intgroup="Condition")
plotCounts(StarDeseq, gene="ENSG00000188817", intgroup="Condition")
plotCounts(StarDeseq, gene="ENSG00000168658", intgroup="Condition")
```



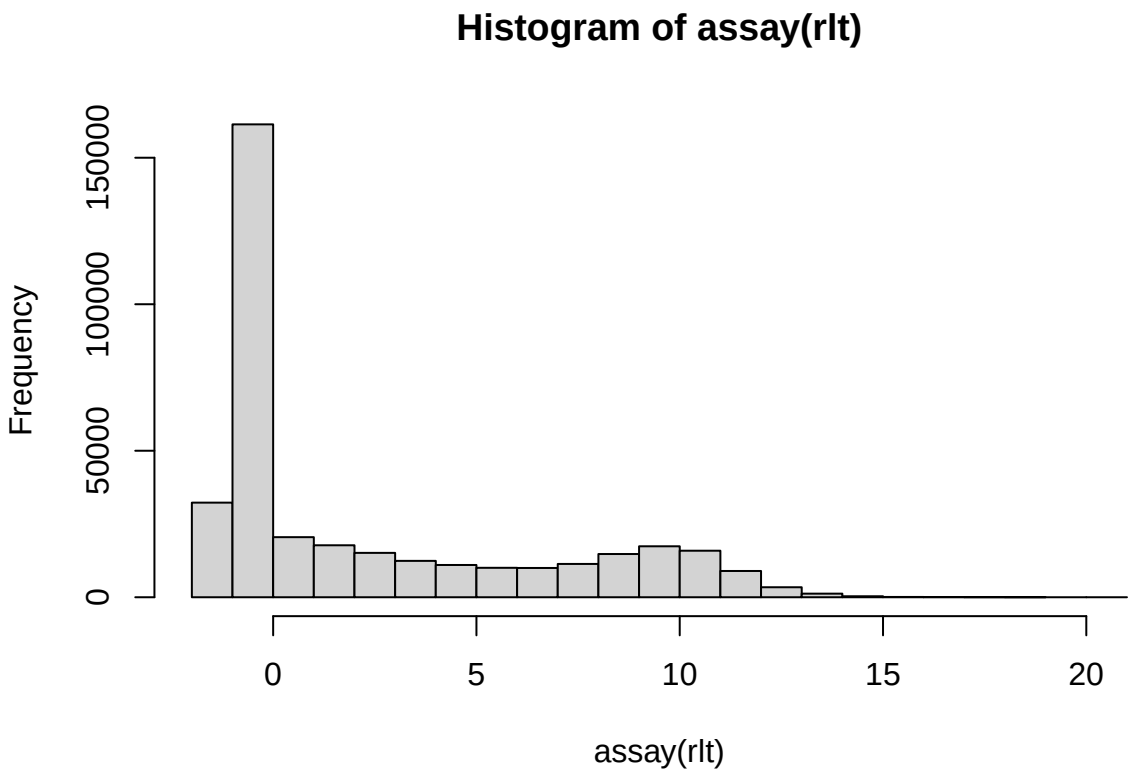

```r
# Principal Component Analysis
#The raw count data is first transformed using the vst function variance stabilizing transformation (vs
vst.data <- vst(StarDeseq, blind=FALSE)
plotPCA(vst.data, intgroup="Condition")
```



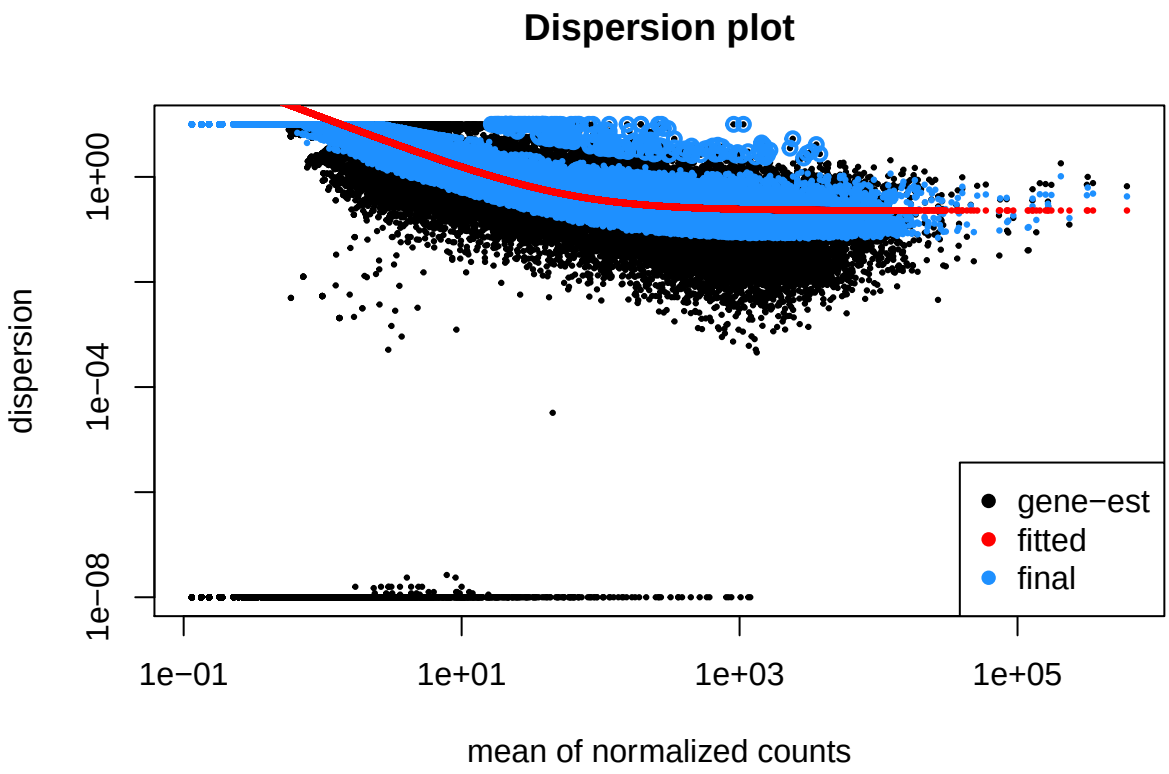

```
# Regularized log transformation for clustering/heatmaps, etc
rlt <- rlogTransformation(StarDeseq)
head(assay(rlt)) #Assay is a summarized experiment matrix like container with rows as features of inter
```

```
##                 sample37 sample38 sample39 sample40 sample41 sample42
## ENSG00000223972 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
## ENSG00000227232 6.922080 6.379790 7.169270 6.000865 6.205120 5.917766
## ENSG00000278267 3.381609 3.904719 4.638531 2.361908 2.220658 2.856369
## ENSG00000243485 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
## ENSG00000284332 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
## ENSG00000237613 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```

```
par(mfrow=c(1,1))
hist(assay(rlt))
```




```
#Plot dispersions
par(mfrow=c(1,1))
#png("qc-dispersions.png", 1000, 1000, pointsize=20)
plotDispEsts(StarDeseq, main="Dispersion plot")
```



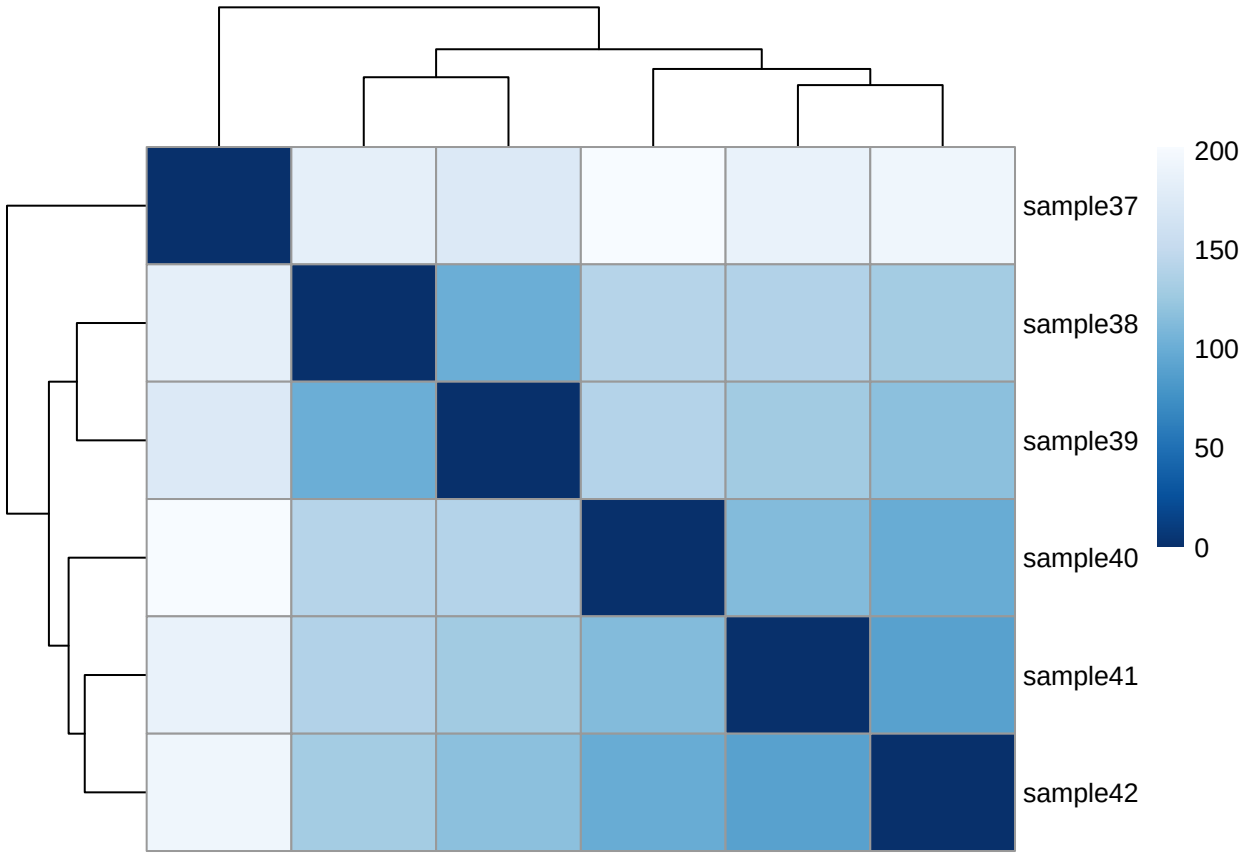
Blind Dispersion Estimation and Extracting Transformed Values

DESeq has two functions `vst` and `rlog` that have an argument *blind* for whether the transformation should be blind to the sample information specified by the design formula. When blind equals TRUE (the default), the functions will re-estimate the dispersions using only an intercept. This setting should be used in order to compare samples in a manner wholly unbiased by the information about experimental groups.
However, when one expects that many or the majority of genes (rows) will have large differences in counts which are explainable by the experimental design, and one wants to transform the data for downstream analysis, blind dispersion is not an appropriate choice. This is because it will lead to large estimations of dispersion.
The running times are shorter when using blind=FALSE and if the function DESeq has already been run, because then it is not necessary to re-estimate the dispersion values. The assay function is used to extract the matrix of normalized values.

```
vsd <- vst(StarDeseq, blind=FALSE)
rld <- rlog(StarDeseq, blind=FALSE)
head(assay(vsd), 3)
```

```
##                 sample37 sample38 sample39 sample40 sample41 sample42
## ENSG00000223972 5.092498 5.092498 5.092498 5.092498 5.092498 5.092498
## ENSG00000227232 7.682696 7.146647 7.939780 6.804078 6.985167 6.733020
## ENSG00000278267 5.932844 6.265407 6.808941 5.328364 5.092498 5.632029
```

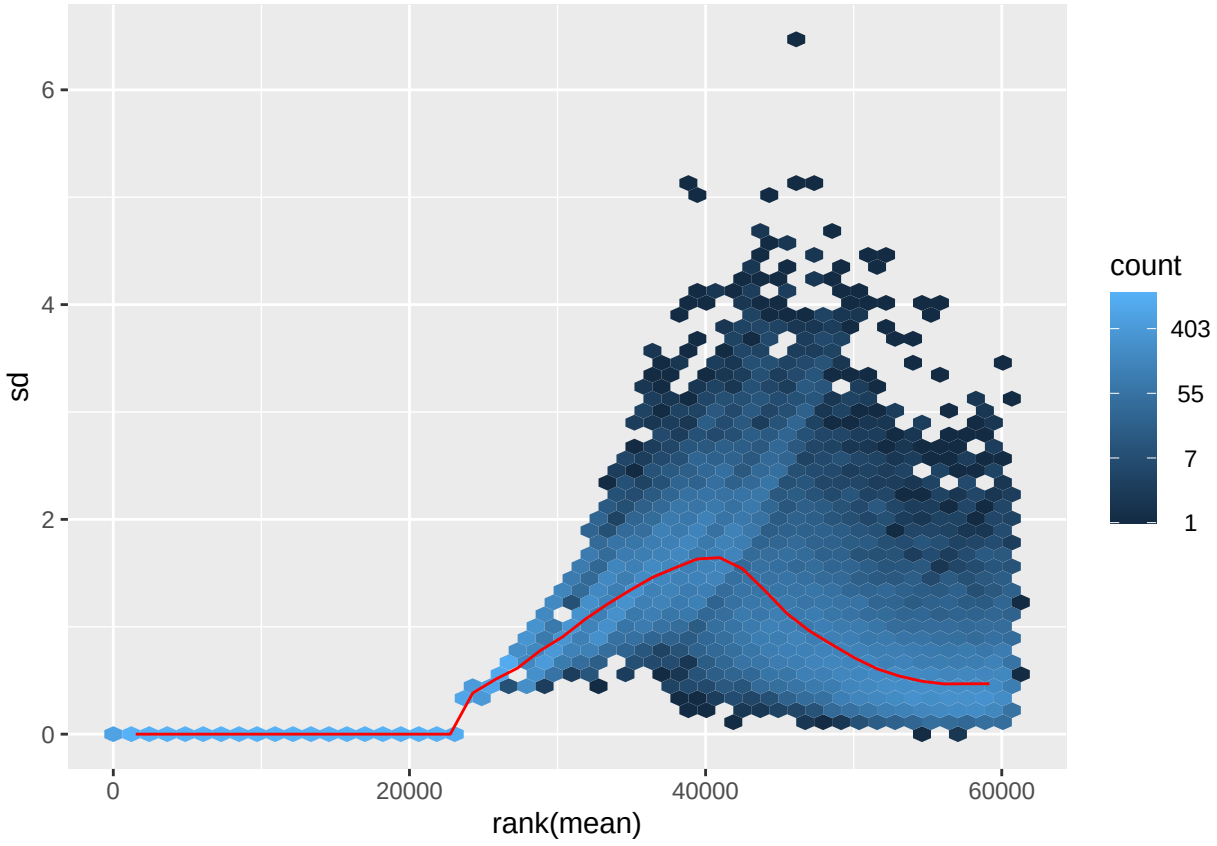
```
# Sample distance heatmap
sampleDists <- dist(t(assay(vsd)))
library(pheatmap)
library("RColorBrewer")
sampleDistMatrix <- as.matrix(sampleDists)
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



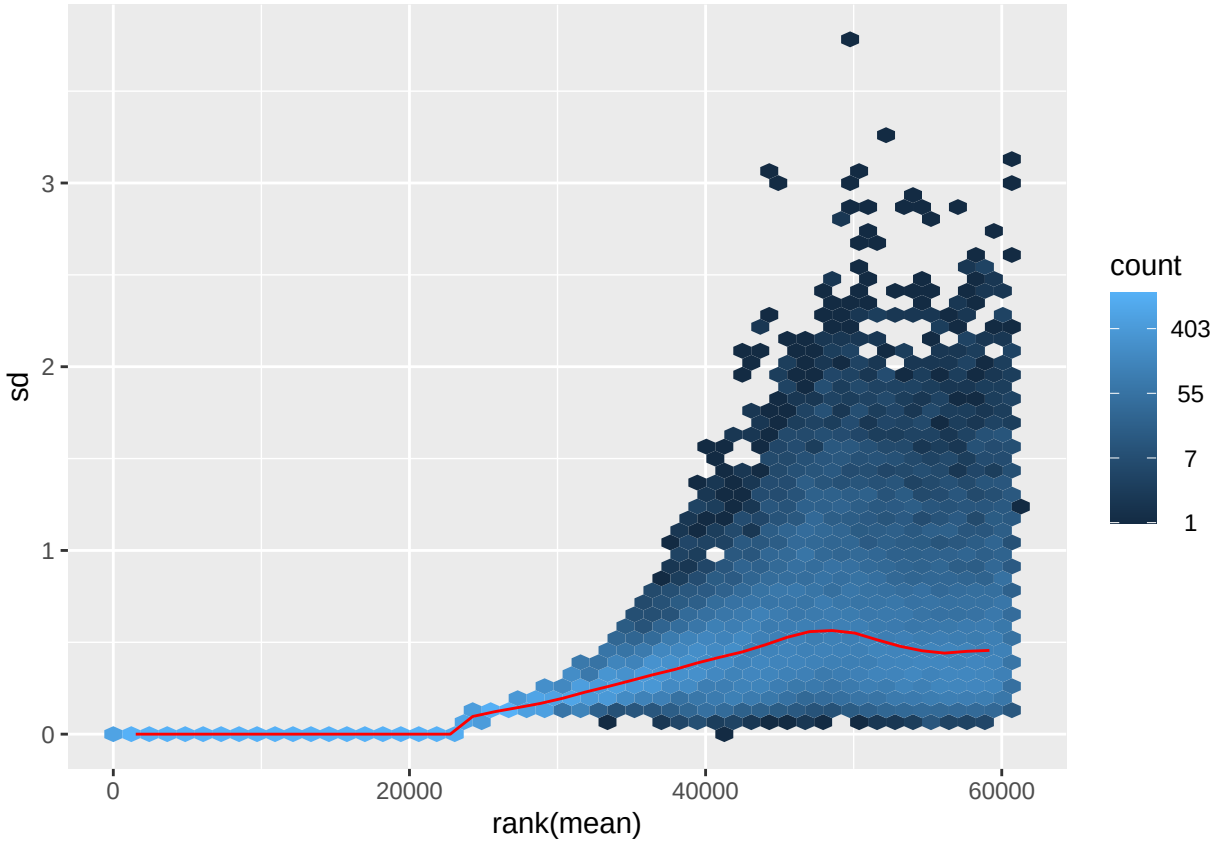
The Effects of Transformation on Variance

The figure below plots the standard deviation of the transformed data, across samples, against the mean, using the shifted logarithm transformation, the regularized log transformation and the variance stabilizing transformation.

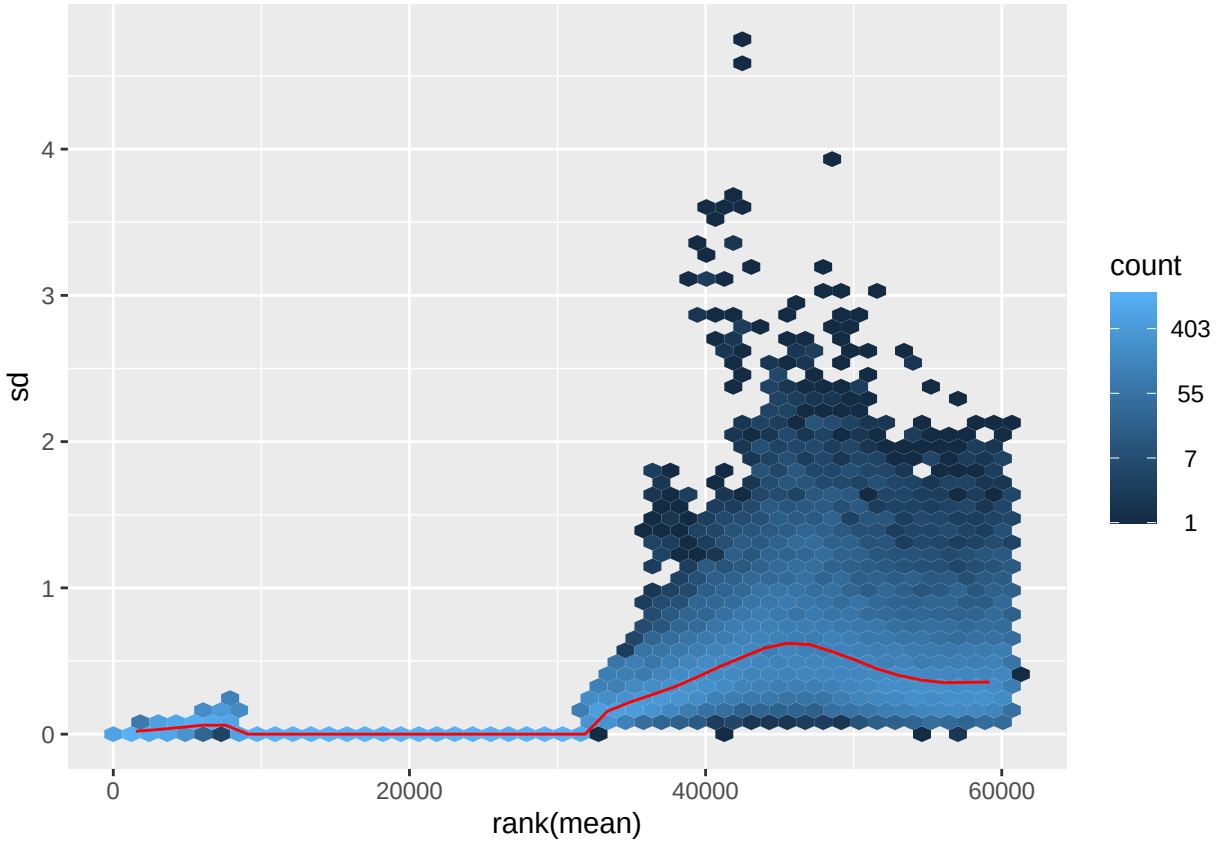
```
trans<- normTransform(StarDeseq)
library("vsn")
meanSdPlot(assay(trans))
```




```
meanSdPlot(assay(vsd))
```



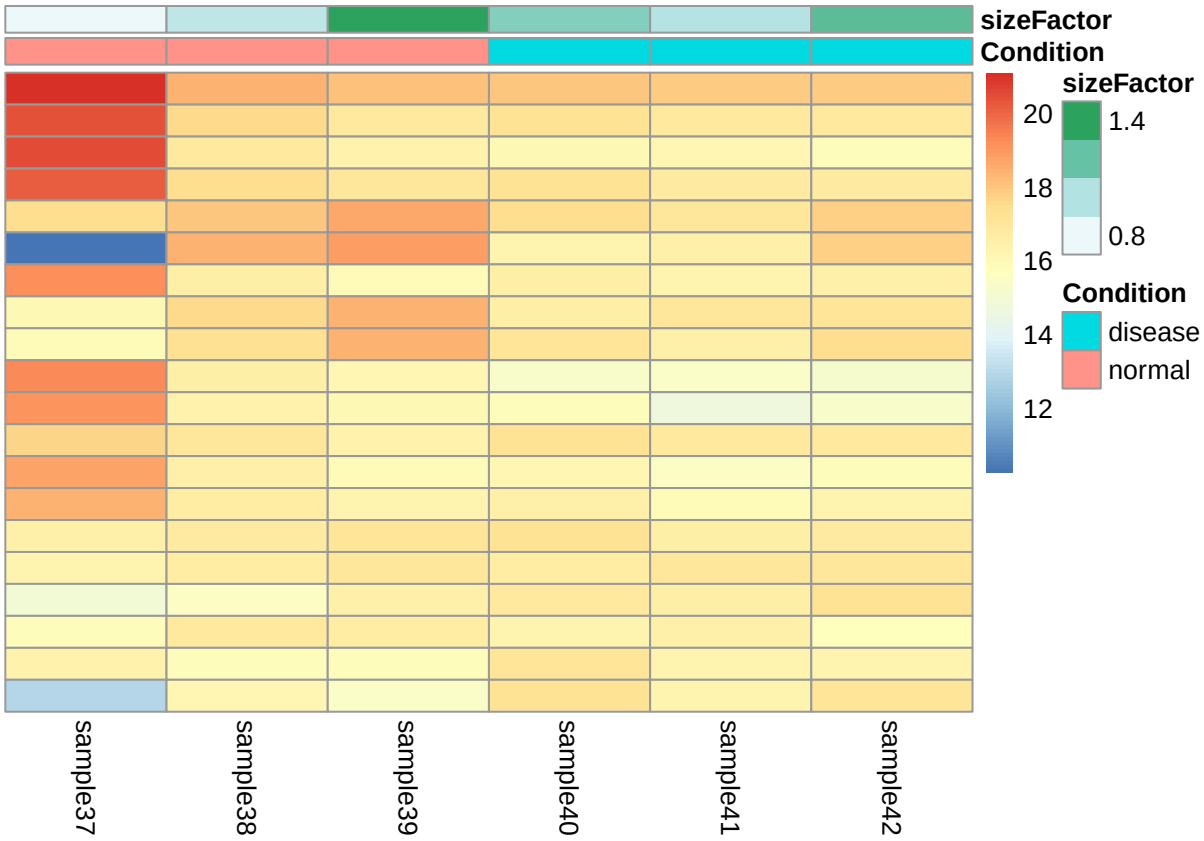

```
meanSdPlot(assay(rld))
```



Sample clustering and visualization for data quality Assessment

Removal of insufficiently good data forms the basis of data quality assessment in data analysis. For this reason, the count matrix generated is instructively looked at as a heatmap as shown below.

```
library("pheatmap")
select <- order(rowMeans(counts(StarDeseq,normalized=TRUE)),
                decreasing=TRUE)[1:20]
df <- as.data.frame(colData(StarDeseq))
pheatmap(assay(trans)[select,], cluster_rows=FALSE, show_rownames=FALSE,
         cluster_cols=FALSE, annotation_col=df)
```



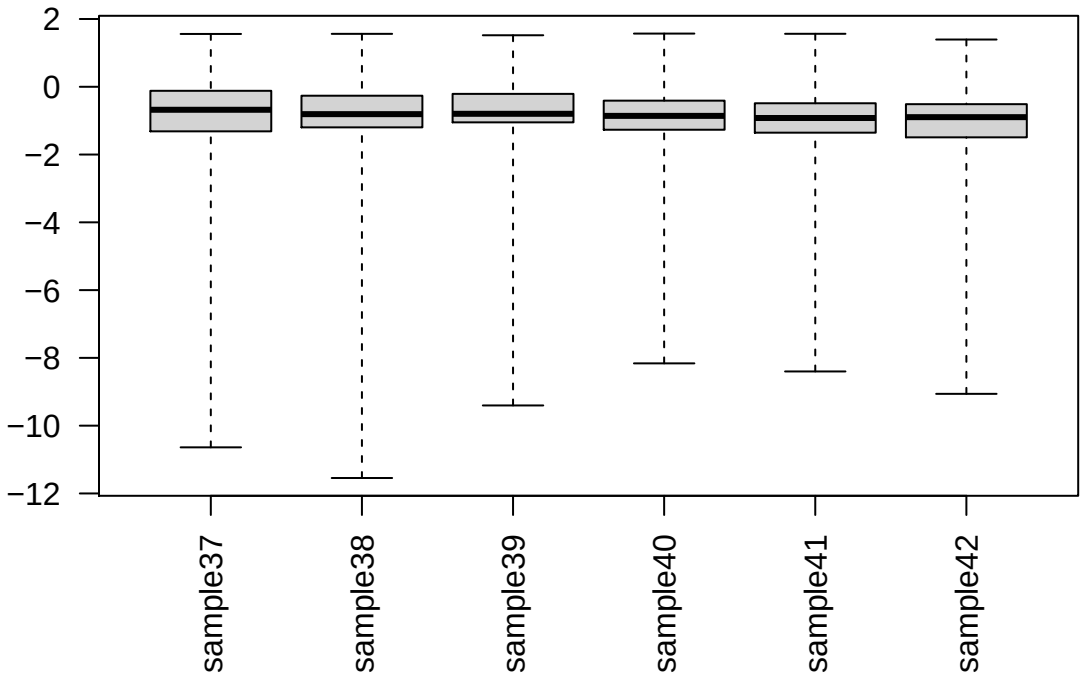
Dealing with outliers

It is possible to have instances of very large counts that are apparently unrelated to the study, and are hence considered as outliers.

A lot of reasons lie behind getting outliers, such as experimental and technical errors, read mapping problems etc.

If there are many outliers it is normally advisable to further explore due to low quality. DESeq calculates, for every gene and for every sample, a diagnostic test for outliers called Cook's distance. Cook's distance is a measure of how much a single sample is influencing the fitted coefficients for a gene, and a large value of Cook's distance is intended to indicate an outlier count. The Cook's distances are stored as a matrix.

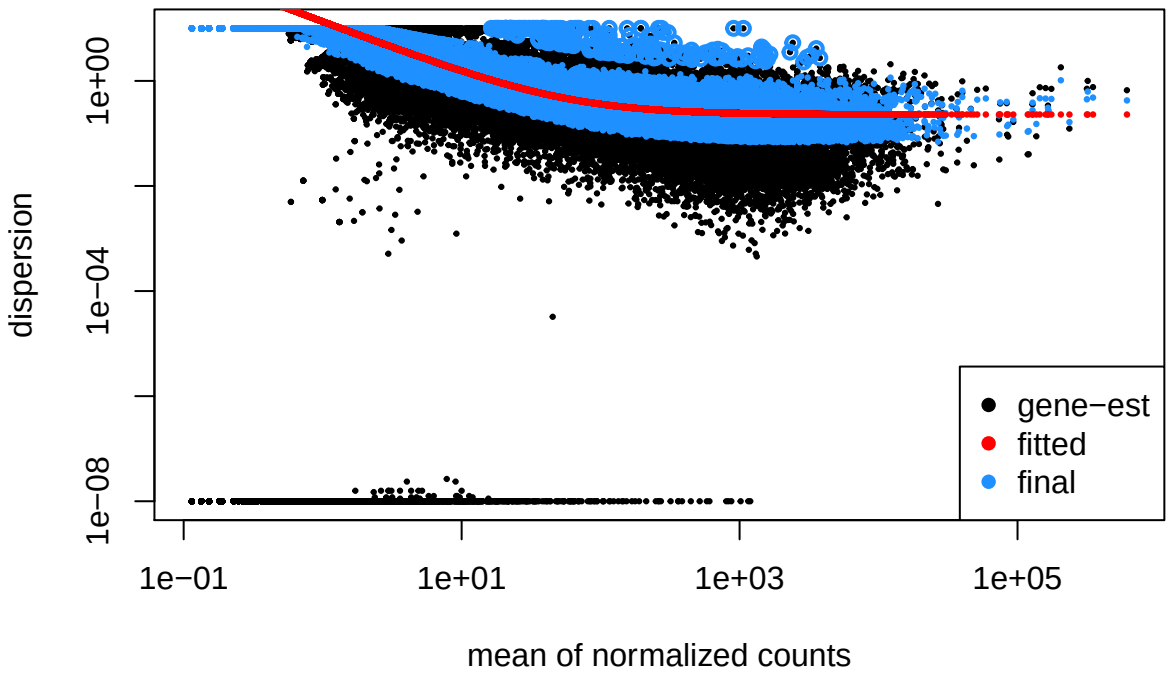
```
#assays(StarDeseq)[["cooks"]]
par(mar=c(8,5,2,2))
boxplot(log10(assays(StarDeseq)[["cooks"]]), range=0, las=2)
```



Dispersion Plots and Fitting Alternatives

Dispersion plots are a useful diagnostic tool for showing final estimates that are shrunk from genewise estimates towards the fitted estimates. However, ome gene-wise estimates are flagged as outliers and not shrunk towards the fitted value.

```
plotDispEsts(StarDeseq)
```



Independent filtering and multiple testing

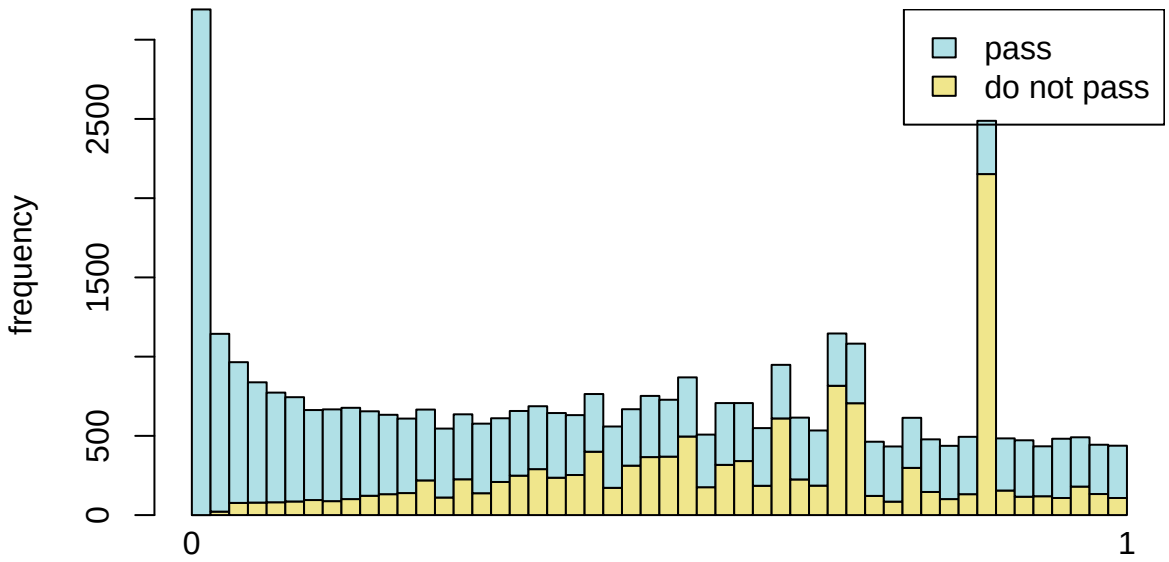
Independent filtering is used to filterout tests from the procedure that have no, or little chance of showing significant evidence without looking at their test statistic.

A good choice for a filtering criterion is one that; 1.Is statistically independent from the test statistic under the null hypothesis, 2.Is correlated with the test statistic under the alternative, and 3.Does not notably change the dependence structure – if there is any – between the tests that pass the filter, compared to the dependence structure between the tests before filtering.

The *p value* histogram below shows how filtering makes the multiple testing strategy better.

```
use <- res$baseMean > metadata(res)$filterThreshold
h1 <- hist(res$pvalue[!use], breaks=0:50/50, plot=FALSE)
h2 <- hist(res$pvalue[use], breaks=0:50/50, plot=FALSE)
colori <- c(`do not pass`="khaki", `pass`="powderblue")

barplot(height = rbind(h1$counts, h2$counts), beside = FALSE,
        col = colori, space = 0, main = "", ylab="frequency")
text(x = c(0, length(h1$counts)), y = 0, label = paste(c(0,1)),
     adj = c(0.5,1.7), xpd=NA)
legend("topright", fill=rev(colori), legend=rev(names(colori)))
```




```
rlt_pca <- function (rlt, intgroup = "Condition", ntop = 500, colors=NULL, legendpos="bottomleft", main=
require(genefilter)
require(calibrate)
require(RColorBrewer)
r.v = rowVars(assay(rlt))
select = order(r.v, decreasing = TRUE)[seq_len(min(ntop, length(r.v)))]
pca = prcomp(t(assay(rlt)[select, ]))
fac = factor(apply(as.data.frame(colData(rlt)[, intgroup, drop = FALSE]), 1, paste, collapse = " : "))
if (is.null(colors)) {
if (nlevels(fac) >= 3) {
colors = brewer.pal(nlevels(fac), "Paired")
} else {
colors = c("black", "red")
}
}
pc1var <- round(summary(pca)$importance[2,1]*100, digits=1)
pc2var <- round(summary(pca)$importance[2,2]*100, digits=1)
pc1lab <- paste0("PC1 (",as.character(pc1var),"%)")
pc2lab <- paste0("PC1 (",as.character(pc2var),"%)")
plot(PC2~PC1, data=as.data.frame(pca$x), bg=colors[fac], pch=21, xlab=pc1lab, ylab=pc2lab, main=main, .
with(as.data.frame(pca$x), textxy(PC1, PC2, labs=rownames(as.data.frame(pca$x)), cex=textcx))
legend(legendpos, legend=levels(fac), col=colors, pch=20)

}
png("qc-pca.png", 1000, 1000, pointsize=20)
rlt_pca(rlt, colors=mycols, intgroup="Condition", xlim=c(-75, 35))
```

```
## Loading required package: genefilter

##
## Attaching package: 'genefilter'

## The following objects are masked from 'package:matrixStats':
##
##     rowSds, rowVars

## Loading required package: calibrate

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:genefilter':
##
##     area
```

Alternative Shrinkage Estimators

DESeq2 contains additional shrinkage estimators besides those moderated log fold changes proposed by Love & Huber, 2014 that use a normal prior distribution, centered on zero and with a scale that is fit to the data. The shrunken log fold changes are useful for ranking and visualization, without the need for arbitrary filters on low count genes. The normal prior at times produces shrinkage that is too strong for certain datasets.

The additional adaptive shrinkage estimators used by DESeq2 are available via the `type` argument of `lfcShrink`. They include `apeglm` and `ashr`. `normal` is the the original DESeq2 shrinkage estimator, it employs an adaptive Normal distribution as prior.

```
resultsNames(StarDeseq)
```

```
## [1] "Intercept"                   "Condition_normal_vs_disease"
```

```
library(apeglm)
library(ashr)
resLFC <- lfcShrink(StarDeseq, coef="Condition_normal_vs_disease", type="apeglm")
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

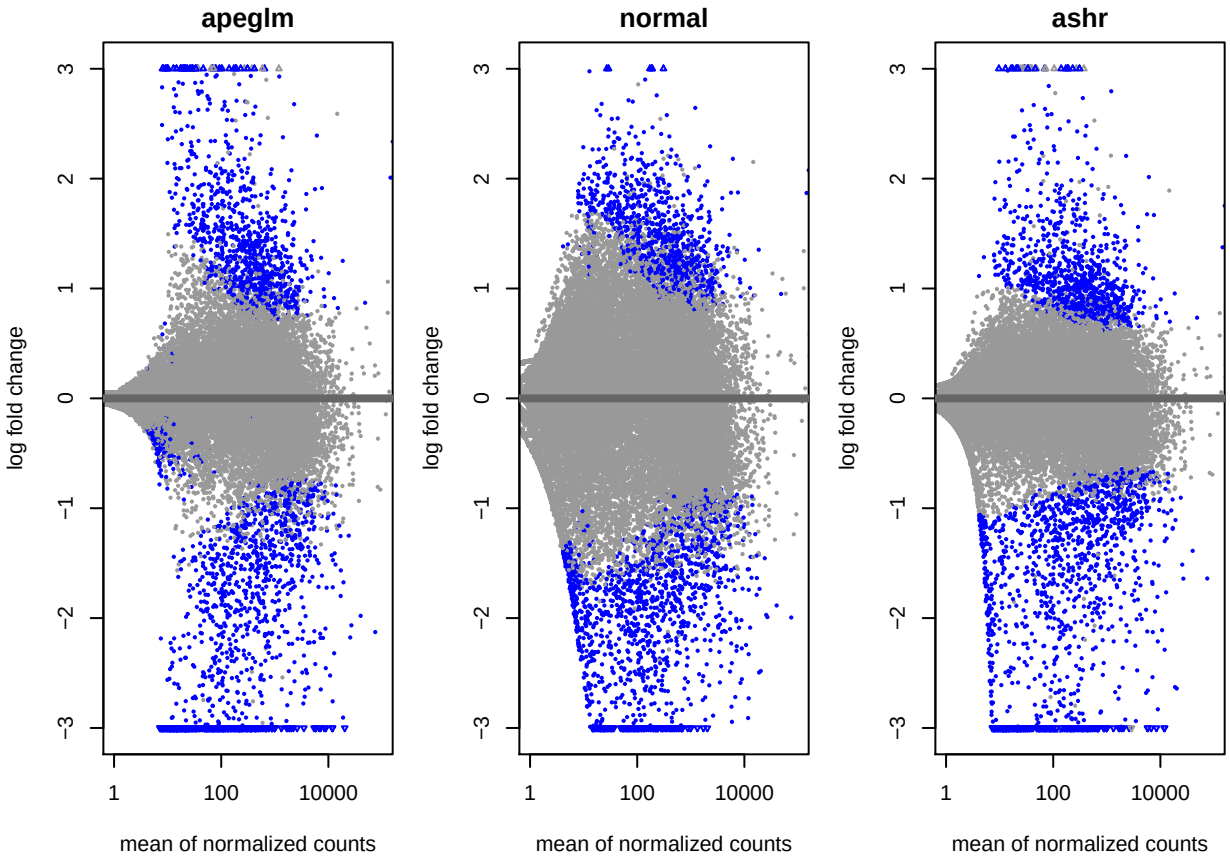
```
resNorm <- lfcShrink(StarDeseq, coef=2, type="normal")
```

```
## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

```
resAsh <- lfcShrink(StarDeseq, coef=2, type="ashr")
```

```
## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041
```

```
par(mfrow=c(1,3), mar=c(4,4,2,1))
xlim <- c(1,1e5); ylim <- c(-3,3)
plotMA(resLFC, xlim=xlim, ylim=ylim, main="apeglm")
plotMA(resNorm, xlim=xlim, ylim=ylim, main="normal")
plotMA(resAsh, xlim=xlim, ylim=ylim, main="ashr")
```



Getting differential Expression Results

```
res2 <- results(StarDeseq)
table(res2$padj<0.05)
```

```
##
## FALSE  TRUE
## 22817  1601
```

```
## Order by adjusted p-value
res2 <- res2[order(res2$padj), ]
## Merge with normalized count data
resdata <- merge(as.data.frame(res2), as.data.frame(counts(StarDeseq, normalized=TRUE)), by="row.names"
names(resdata)[1] <- "Gene"
head(resdata)
```

```
##              Gene  baseMean log2FoldChange     lfcSE      stat       pvalue
## 1 ENSG00000039537  432.6920      -6.929060 0.7283064 -9.513935 1.835836e-21
## 2 ENSG00000124237  241.9873      -7.992429 0.8563060 -9.333613 1.023224e-20
## 3 ENSG00000160401  508.1132      -6.040486 0.6572527 -9.190507 3.909942e-20
## 4 ENSG00000007908 1329.3382      -6.212179 0.6904355 -8.997479 2.309610e-19
## 5 ENSG00000188817  409.4163      -5.830750 0.6657424 -8.758267 1.982750e-18
## 6 ENSG00000168658  330.1829     -11.738757 1.3967993 -8.404040 4.313755e-17
##           padj  sample37  sample38  sample39 sample40  sample41  sample42
## 1 4.482744e-17  0.000000 13.612156  6.869339 1097.705  557.4970  920.4684
## 2 1.249254e-16  0.000000  2.268693  2.747736  718.398  261.8381  466.6710
## 3 3.182432e-16  8.930348 24.955620 11.677877 1472.442  421.1230 1109.5506
## 4 1.409902e-15 56.558871 27.224313 23.355753 3094.778 4013.7601  760.3519
## 5 9.682960e-15 11.907131 22.686927  8.243207 1149.802  327.2976  936.5605
## 6 1.659971e-13  0.000000  0.000000  0.000000 1022.757  196.3786  761.9612
```

```
## Write results
write.csv(resdata, file="diffexpr-results.csv")


#MAPLOT
maplot <- function (res2, thresh=0.05, labelsig=TRUE, textcx=1, ...) {
with(res2, plot(baseMean, log2FoldChange, pch=20, cex=.5, log="x", ...))
with(subset(res2, padj<thresh), points(baseMean, log2FoldChange, col="red", pch=20, cex=1.5))
if (labelsig) {
require(calibrate)
with(subset(res2, padj<thresh), textxy(baseMean, log2FoldChange, labs=Gene, cex=textcx, col=2))
}
}
png("diffexpr-maplot.png", 1500, 1000, pointsize=20)
ma.plt <- maplot(resdata, main="MA Plot")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 23392 x values <= 0 omitted
## from logarithmic plot
```

```
ma.plt
```

```
## NULL
```

Adding Annotation to DESeq2 Results

After obtaining a list of differentially expressed genes, the only annotation present is normally the Ensembl Gene ID, which is not as informative.
This can be done using `org.Mm.eg.db` package or `biomaRt`, an interface to the biomaRt resource.
The `org.Hs.eg.db` package is the organism annotation package for humans organized as an AnnotationDbi database package ("db"), using Entrez Gene IDs ("eg") as primary key.

```
library("AnnotationDbi")
```

```
##
## Attaching package: 'AnnotationDbi'

## The following object is masked from 'package:MASS':
##
##     select
```

```
library("org.Hs.eg.db")
```

```
##
```

```
#To get a list of available types:
columns(org.Hs.eg.db)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GO"           "GOALL"        "IPI"          "MAP"          "OMIM"
## [16] "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"         "PMID"
## [21] "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"       "UNIGENE"
## [26] "UNIPROT"
```

```
#Converting native functions from the annotationDbi package

convertIDs <- function( ids, from, to, db, ifMultiple=c("putNA", "useFirst")) {
  stopifnot( inherits( db, "AnnotationDb" ) )
  ifMultiple <- match.arg( ifMultiple )
  suppressWarnings( selRes <- AnnotationDbi::select(
    db, keys=ids, keytype=from, columns=c(from,to) ) )
  if ( ifMultiple == "putNA" ) {
    duplicatedIds <- selRes[ duplicated( selRes[,1] ), 1 ]
    selRes <- selRes[ ! selRes[,1] %in% duplicatedIds, ]
  }
  return( selRes[ match( ids, selRes[,1] ), 2 ] )
}
```

The function takes a list of IDs as first argument and their key type as the second argument. The third argument is the key type we want to convert to, the fourth is the AnnotationDb object to use. Finally, the last argument specifies what to do if one source ID maps to several target IDs, whether the function return an NA or simply the first of the multiple IDs? To convert the Ensembl IDs in the rownames of `res` to gene symbols and add them as a new column,the function below was used:

```
res$hgnc_symbol <- convertIDs(row.names(res), "ENSEMBL", "SYMBOL", org.Hs.eg.db)
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
res$entrezgene <- convertIDs(row.names(res), "ENSEMBL", "ENTREZID", org.Hs.eg.db)
```

```
## 'select()' returned 1:many mapping between keys and columns
```

Once the desired external geneIDs have been attached to the results:

```
resordered <- res[order(res$pvalue),]
head(resordered)
```

```
## log2 fold change (MLE): Condition normal vs disease
## Wald test p-value: Condition normal vs disease
## DataFrame with 6 rows and 8 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000039537   432.692       -6.92906  0.728306  -9.51394 1.83584e-21
## ENSG00000124237   241.987       -7.99243  0.856306  -9.33361 1.02322e-20
## ENSG00000160401   508.113       -6.04049  0.657253  -9.19051 3.90994e-20
## ENSG00000007908  1329.338       -6.21218  0.690436  -8.99748 2.30961e-19
## ENSG00000188817   409.416       -5.83075  0.665742  -8.75827 1.98275e-18
## ENSG00000168658   330.183      -11.73876  1.396799  -8.40404 4.31376e-17
##                        padj hgnc_symbol  entrezgene
##                   <numeric> <character> <character>
## ENSG00000039537 4.48274e-17          C6         729
## ENSG00000124237 1.24925e-16    C20orf85      128602
## ENSG00000160401 3.18243e-16     CFAP157      286207
## ENSG00000007908 1.40990e-15        SELE        6401
## ENSG00000188817 9.68296e-15        SNTN      132203
## ENSG00000168658 1.65997e-13       VWA3B      200403
```

The results can then be exported as shown below:

```
write.csv(as.data.frame(resordered), file="results.csv")
```