DESeq2_analysis of the SRP033351 data

SP:BITS

April 27, 2015 / update June 16, 2020

All preliminary steps were performed in separate training exercises. We have at this point HTSeq counts for each sample and continue with the DESeq2 vignette http://www.bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.pdf where the HTSeq data is loaded in a DESeq2 object for analysis.

Required packages

```
library("DESeq2")
library("ggplot2")
library("vsn")
library("RColorBrewer")
library("gplots")
library("openxlsx")
```

Locate and load data

First we want to specify a variable which points to the directory in which the HTSeq output files are located. We then create a metadata table that will help ordering and merging the results.

```
# please adapt this location to match your own environment
basedir <- "/media/bits/RNASeq_DATA"
basedir <- "/Users/u0002316/Desktop/GSE52778"
setwd(basedir)

# load metadata
metadata <- read.table(paste(basedir, "GSE52778_metadata.txt", sep="/"), header = TRUE)
metadata$sampleFiles <- paste( metadata$run_accession, "_all_counts.txt", sep="")

# restrict to untreated and Dex samples
selectedRows <- metadata[grep("untreated|^Dex", metadata$treatment), ]

sampleTable <- data.frame(sampleName = selectedRows$run_accession,
  fileName = selectedRows$sampleFiles,
  cells = selectedRows$cells,
  treatment = selectedRows$treatment)

sampleTable
```

```
##   sampleName                  fileName   cells treatment
## 1 SRR1039508 SRR1039508_all_counts.txt  N61311 untreated
## 2 SRR1039509 SRR1039509_all_counts.txt  N61311       Dex
## 3 SRR1039512 SRR1039512_all_counts.txt N052611 untreated
## 4 SRR1039513 SRR1039513_all_counts.txt N052611       Dex
## 5 SRR1039516 SRR1039516_all_counts.txt N080611 untreated
## 6 SRR1039517 SRR1039517_all_counts.txt N080611       Dex
## 7 SRR1039520 SRR1039520_all_counts.txt N061011 untreated
## 8 SRR1039521 SRR1039521_all_counts.txt N061011       Dex
```

HTSeq input

Load HTSeq data into a DESeq2 object.

```
# point directory to the folder containing the htseq count files
# directory <- "/work/TUTORIALS/NGS_RNASeqDE-training2015/htseq_counts"
directory <- paste(basedir, "htseq_counts", sep="/")

dds <- DESeqDataSetFromHTSeqCount(sampleTable = sampleTable,
                                  directory = directory,
                                  design = ~ cells + treatment)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
# review the object
dds
```

```
## class: DESeqDataSet
## dim: 57773 8
## metadata(1): version
## assays(1): counts
## rownames(57773): ENSG00000000003 ENSG00000000005 ... ENSG00000273492
##   ENSG00000273493
## rowData names(0):
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(2): cells treatment
```

```
# column information
colData(dds)
```

```
## DataFrame with 8 rows and 2 columns
##               cells treatment
##            <factor>  <factor>
## SRR1039508  N61311  untreated
## SRR1039509  N61311  Dex
## SRR1039512  N052611 untreated
## SRR1039513  N052611 Dex
## SRR1039516  N080611 untreated
## SRR1039517  N080611 Dex
## SRR1039520  N061011 untreated
## SRR1039521  N061011 Dex
```

```
# relevel to get untreated as reference
dds$treatment <- relevel(dds$treatment, "untreated")
```

Differential expression analysis

The DESeq function is a wrapper that performs a default analysis through three steps:

- estimation of size factors: estimateSizeFactors
- estimation of dispersion: estimateDispersions
- Negative Binomial GLM fitting and Wald statistics: nbinomWaldTest

```
# run the combined DESeq2 analysis
dds <- DESeq(dds, betaPrior=TRUE)
```

```
## estimating size factors
```

```
## estimating dispersions
```

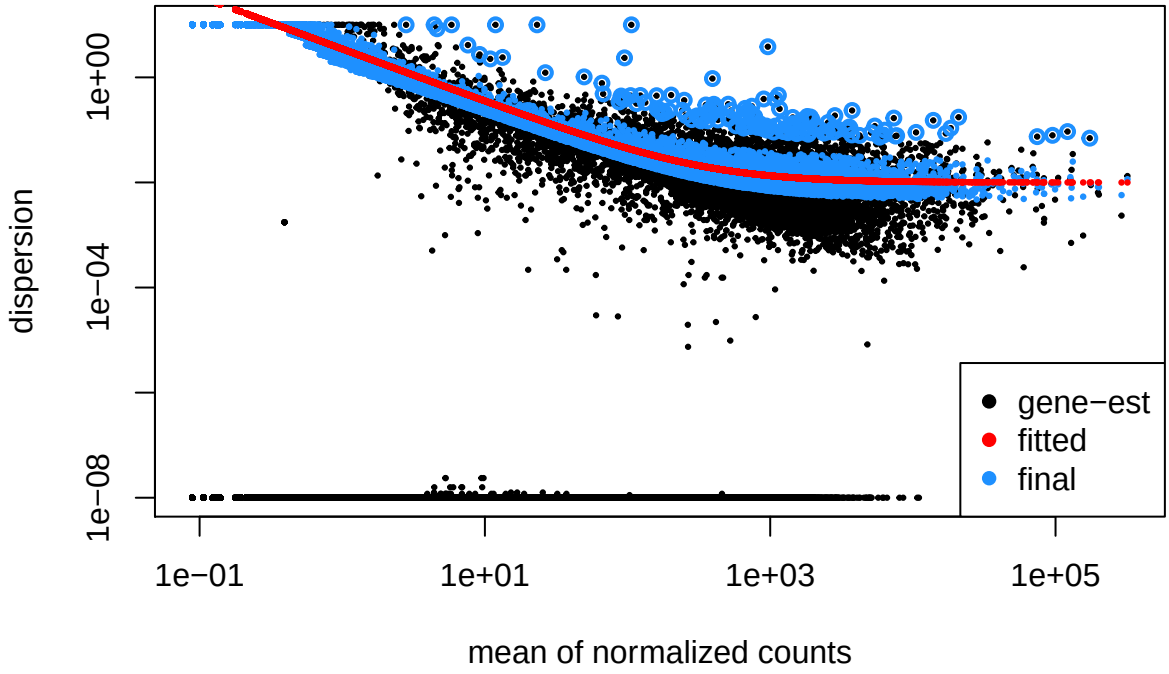
```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

fitting model and testing

```
# estimating size factors
# estimating dispersions
# gene-wise dispersion estimates
# mean-dispersion relationship
# final dispersion estimates
# fitting model and testing

# Dispersion plot and fitting alternatives
plotDispEsts(dds)
```




```
# get size factors used for normalization
sizeFactors(dds)
```

```
## SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
##  1.0123484  0.8951975  1.1747041  0.6739964  1.1729844  1.4049307  0.9193101
## SRR1039521
##  0.9527669
```

```
# store results into a new object
res <- results(dds)
head(res)
```

```
## log2 fold change (MAP): treatment Dex vs untreated
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003 698.925490     -0.3880750  0.100219 -3.872259 0.000107831
## ENSG00000000005   0.000000             NA        NA        NA          NA
## ENSG00000000419 475.485318      0.2112264  0.113531  1.860521 0.062811801
## ENSG00000000457 251.901495      0.0136797  0.133460  0.102500 0.918359508
## ENSG00000000460  50.847560     -0.0268542  0.263189 -0.102034 0.918729665
## ENSG00000000938   0.212976     -0.0558523  0.154690 -0.361060 0.718054440
##                       padj
##                  <numeric>
## ENSG00000000003 0.00092393
## ENSG00000000005         NA
```

```
## ENSG00000000419 0.18895268
## ENSG00000000457 0.96459567
## ENSG00000000460 0.96479354
## ENSG00000000938         NA
```

```
# reorder the results by decreasing significance
resOrdered <- res[order(res$padj),]

# inspect
head(resOrdered)
```

```
## log2 fold change (MAP): treatment Dex vs untreated
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat       pvalue
##                  <numeric>      <numeric> <numeric> <numeric>    <numeric>
## ENSG00000165995   514.284        3.32076  0.130973   25.3546 8.00568e-142
## ENSG00000120129  3325.403        2.87348  0.116269   24.7139 7.57320e-135
## ENSG00000152583   985.559        4.34018  0.175930   24.6700 2.24572e-134
## ENSG00000101347 13616.935        3.60737  0.151117   23.8713 6.08347e-126
## ENSG00000189221  2294.730        3.23216  0.139179   23.2231 2.65891e-119
## ENSG00000211445 12162.487        3.54142  0.156706   22.5991 4.42010e-113
##                         padj
##                    <numeric>
## ENSG00000165995 1.37121e-137
## ENSG00000120129 6.48569e-131
## ENSG00000152583 1.28216e-130
## ENSG00000101347 2.60494e-122
## ENSG00000189221 9.10835e-116
## ENSG00000211445 1.26179e-109
```

```
# We can summarize some basic tallies using the summary function.
summary(res)
```

```
##
## out of 28730 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 2467, 8.6%
## LFC < 0 (down)     : 2155, 7.5%
## outliers [1]       : 0, 0%
## low counts [2]     : 11602, 40%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

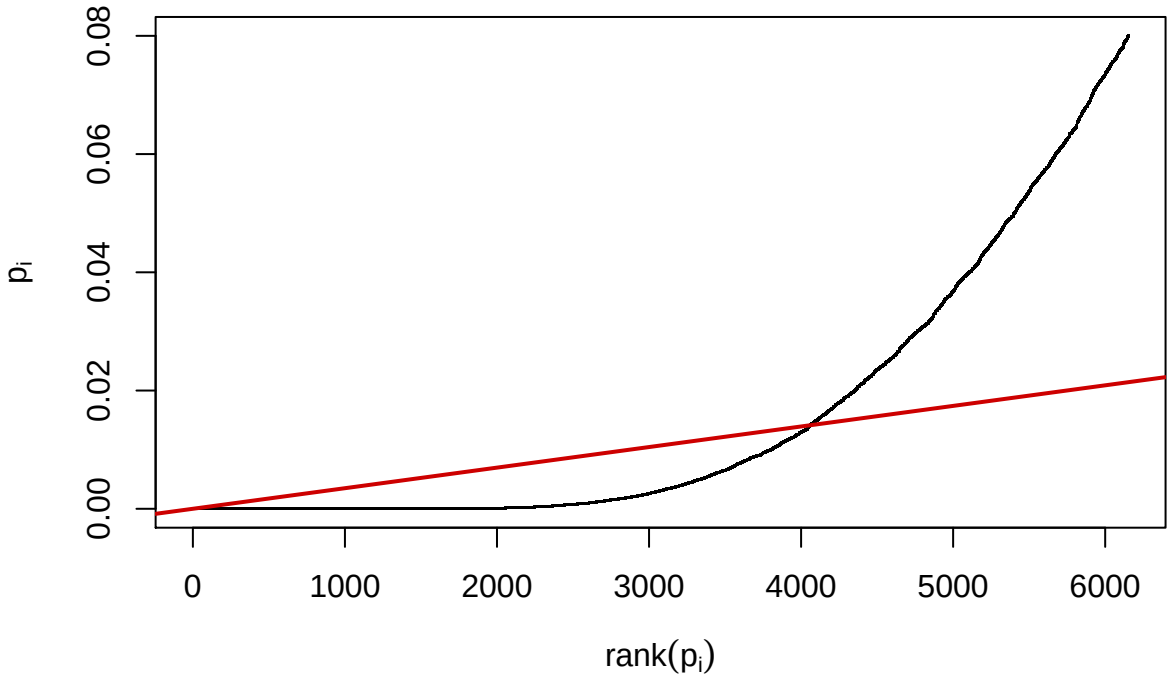
```
## being more stringent
summary(res, alpha=0.01)
```

```
##
## out of 28730 with nonzero total read count
## adjusted p-value < 0.01
## LFC > 0 (up)       : 1533, 5.3%
## LFC < 0 (down)     : 1257, 4.4%
## outliers [1]       : 0, 0%
## low counts [2]     : 11602, 40%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Diagnostic plots for multiple testing

The plot shows the effect of multiple testing and defines the limit of trust of the results as compared to a random situation.

```
# keep only data with computed pval
resFilt <- res[!is.na(res$pvalue),]

# order data by increasing pval
orderInPlot <- order(resFilt$pvalue)

# filter to show only top
showInPlot <- (resFilt$pvalue[orderInPlot] <= 0.08)

# set significance level for testing
alpha <- 0.1

{
  plot(seq(along=which(showInPlot)), resFilt$pvalue[orderInPlot][showInPlot],
     pch=".", xlab = expression(rank(p[i])), ylab=expression(p[i]))

# add limit
abline(a=0, b=alpha/length(resFilt$pvalue), col="red3", lwd=2)
}
```



Exploring and exporting results

```
# current results
head(res, 4)
```

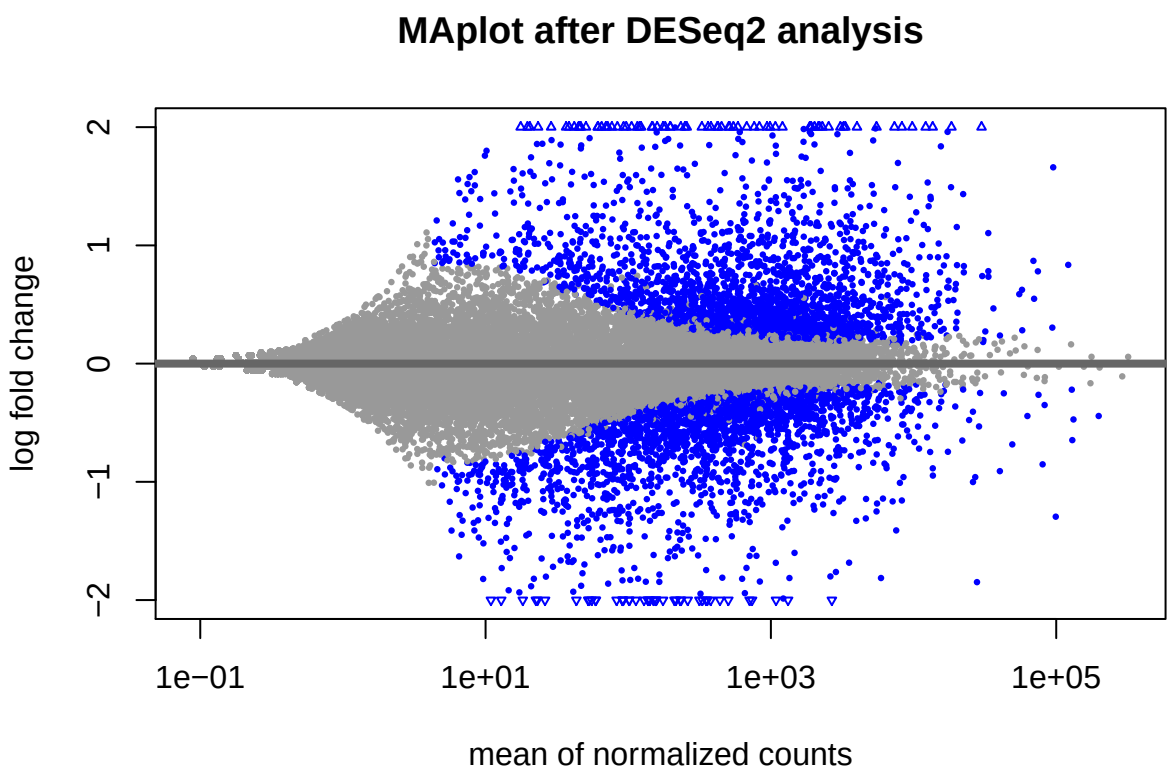
```
## log2 fold change (MAP): treatment Dex vs untreated
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 4 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003   698.925     -0.3880750  0.100219  -3.87226 0.000107831
## ENSG00000000005     0.000             NA        NA        NA          NA
## ENSG00000000419   475.485      0.2112264  0.113531   1.86052 0.062811801
## ENSG00000000457   251.901      0.0136797  0.133460   0.10250 0.918359508
##                       padj
##                  <numeric>
## ENSG00000000003 0.00092393
```

```
## ENSG00000000005         NA
## ENSG00000000419 0.18895268
## ENSG00000000457 0.96459567
```

```
# keep only data with computed pval
resFilt <- res[!is.na(res$pvalue),]

head(resFilt, 4)
```

```
## log2 fold change (MAP): treatment Dex vs untreated
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 4 rows and 6 columns
##                  baseMean log2FoldChange     lfcSE      stat      pvalue
##                 <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003  698.9255     -0.3880750  0.100219 -3.872259 0.000107831
## ENSG00000000419  475.4853      0.2112264  0.113531  1.860521 0.062811801
## ENSG00000000457  251.9015      0.0136797  0.133460  0.102500 0.918359508
## ENSG00000000460   50.8476     -0.0268542  0.263189 -0.102034 0.918729665
##                      padj
##                 <numeric>
## ENSG00000000003 0.00092393
## ENSG00000000419 0.18895268
## ENSG00000000457 0.96459567
## ENSG00000000460 0.96479354
```

```
# MA-plot
# legend: Points will be colored red if the adjusted p value is less than 0.1. Points which fall out of th

# plots for DESeq transformed data
plotMA(res, main="MAplot after DESeq2 analysis", ylim=c(-2,2))
```

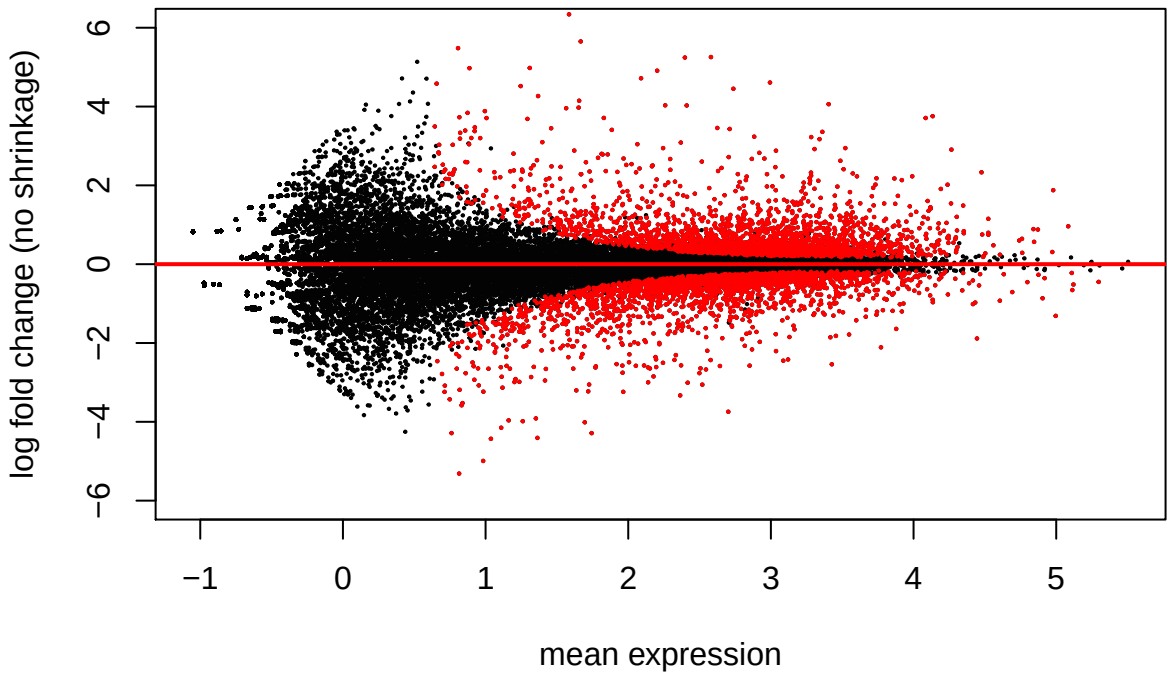



```
# A column lfcMLE with the unshrunken maximum likelihood estimate (MLE) for the log2 fold change will be a
resMLE <- results(dds, addMLE=TRUE)
head(resMLE, 4)
```

```
## log2 fold change (MAP): treatment Dex vs untreated
```

```
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 4 rows and 7 columns
##                   baseMean log2FoldChange     lfcMLE     lfcSE      stat
##                  <numeric>      <numeric>  <numeric> <numeric> <numeric>
## ENSG00000000003   698.925     -0.3880750 -0.3955815  0.100219  -3.87226
## ENSG00000000005     0.000             NA         NA        NA        NA
## ENSG00000000419   475.485      0.2112264  0.2163503  0.113531   1.86052
## ENSG00000000457   251.901      0.0136797  0.0141535  0.133460   0.10250
##                      pvalue       padj
##                   <numeric>  <numeric>
## ENSG00000000003 0.000107831 0.00092393
## ENSG00000000005          NA         NA
## ENSG00000000419 0.062811801 0.18895268
## ENSG00000000457 0.918359508 0.96459567
```

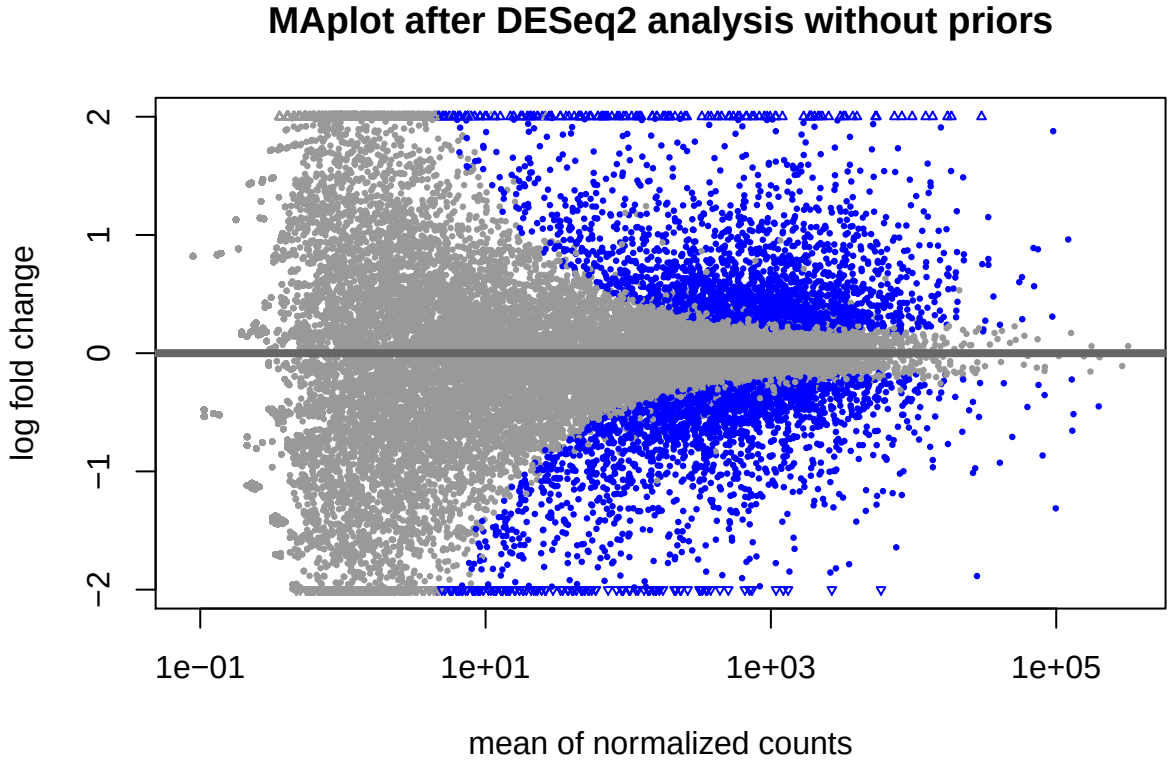
```
# significant calls
# keep only data with computed pval
resMLEFilt <- resMLE[!is.na(resMLE$pvalue),]
resMLESig <- subset(resMLEFilt, resMLEFilt$padj<0.1)

# plotMA with unshrunken values
{
  plot(log(resMLEFilt$baseMean, 10), resMLEFilt$lfcMLE,
     ylim=c(-6,6),
     xlab="mean expression",
     ylab="log fold change (no shrinkage)",
     pch=20,
     col="grey1",
     cex=0.25,
     main="MA-plot without shrinkage (padj<0.1)")

# color significant
points(log(resMLESig$baseMean,10), resMLESig$lfcMLE,
       col="red",
       pch=20,
       cex=0.25)

# add limit
abline(a=0, b=alpha/length(resFilt$pvalue), col="red1", lwd=2)
}
```




```
# SAME using built in capabilities of DESeq2
resNoPrior <- DESeq(dds, betaPrior=FALSE)
```

```
## using pre-existing size factors
## estimating dispersions
## found already estimated dispersions, replacing these
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```

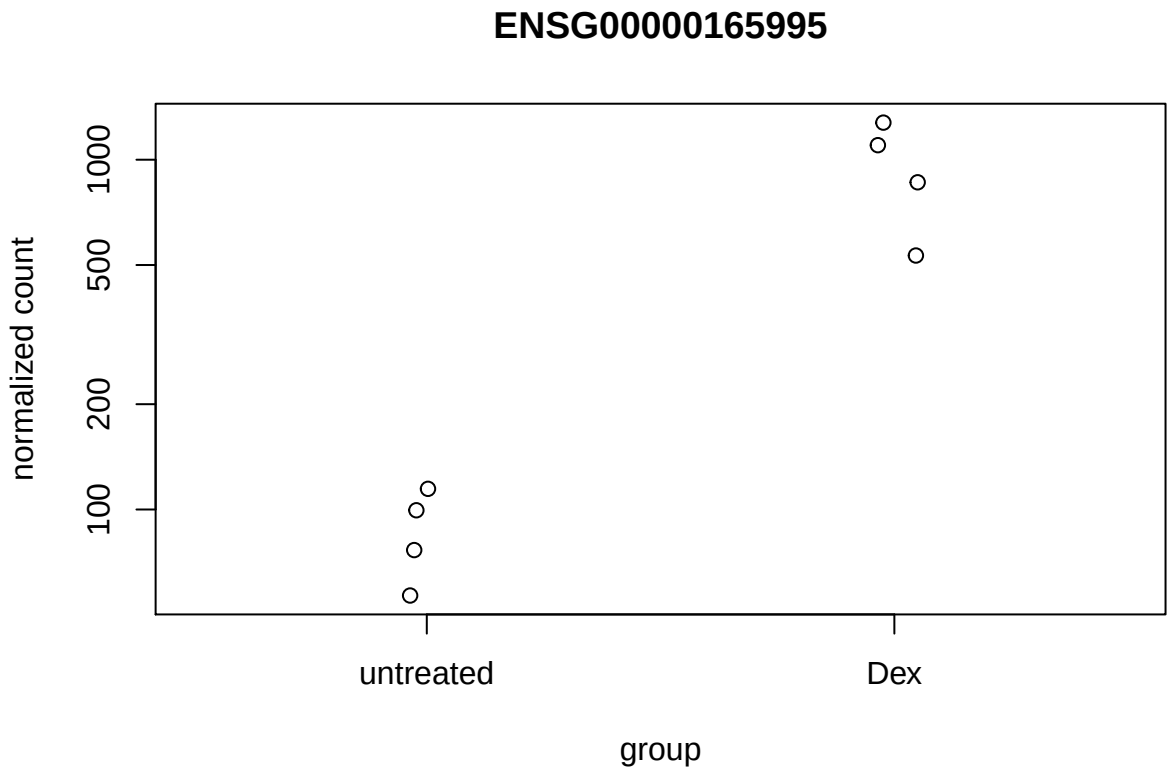
```
plotMA(resNoPrior, main="MAplot after DESeq2 analysis without priors",
       ylim=c(-2,2))
```



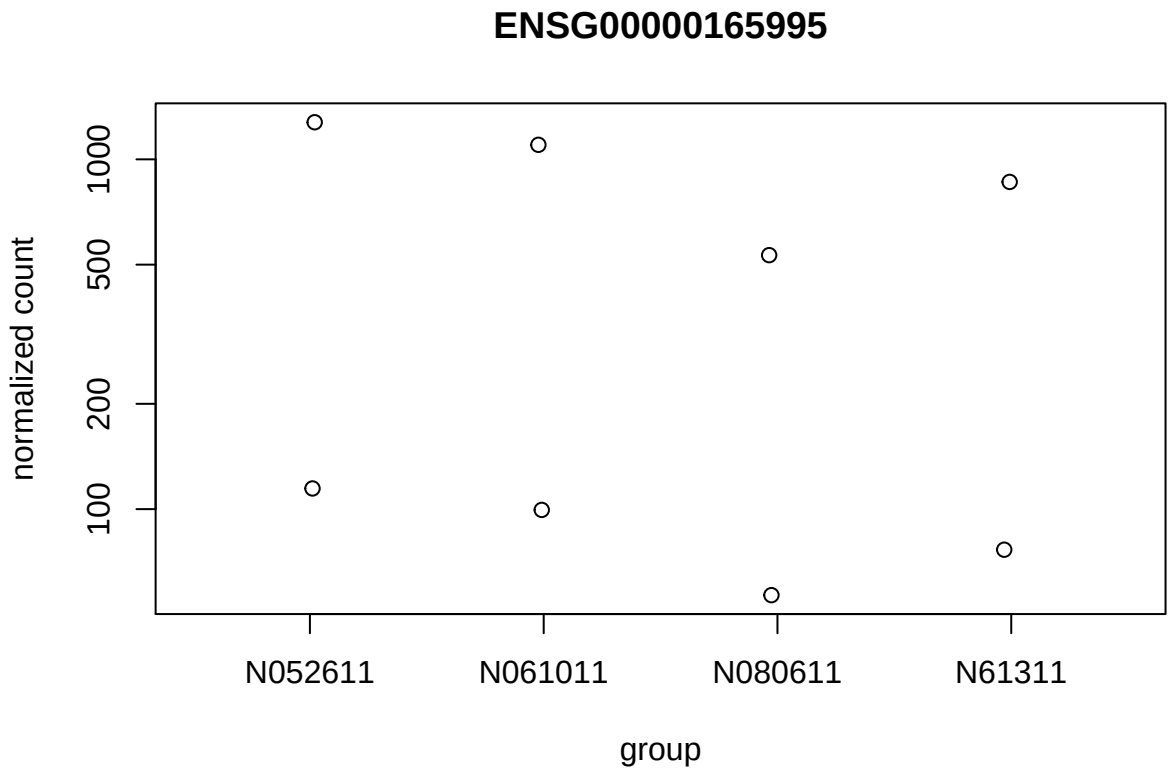
Plot counts

Plot counts for a given gene; here the gene with best pvalue for DE between Dex and untreated.

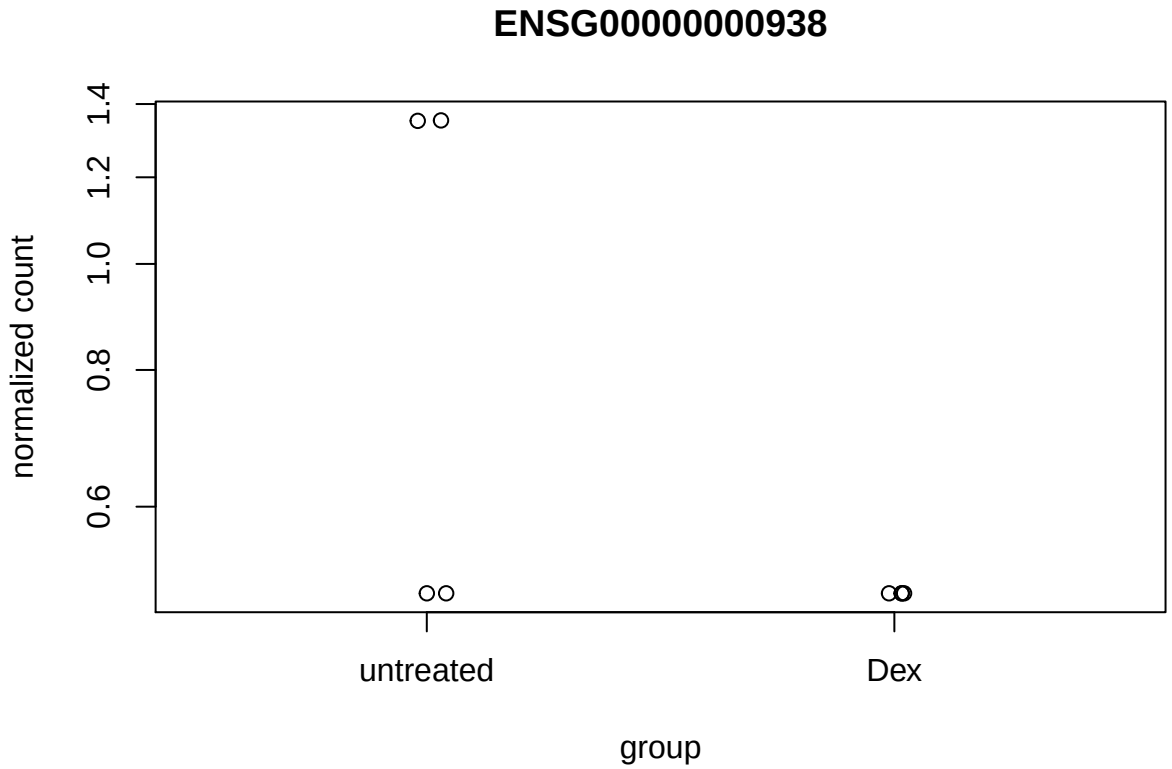
```
# simple plot for the best scoring gene (based on padj)
onegene <- rownames(res)[which.min(res$padj)]

plotCounts(dds, gene=which.min(res$padj), intgroup="treatment", main=onegene)
```



```
plotCounts(dds, gene=which.min(res$padj), intgroup="cells", main=onegene)
```

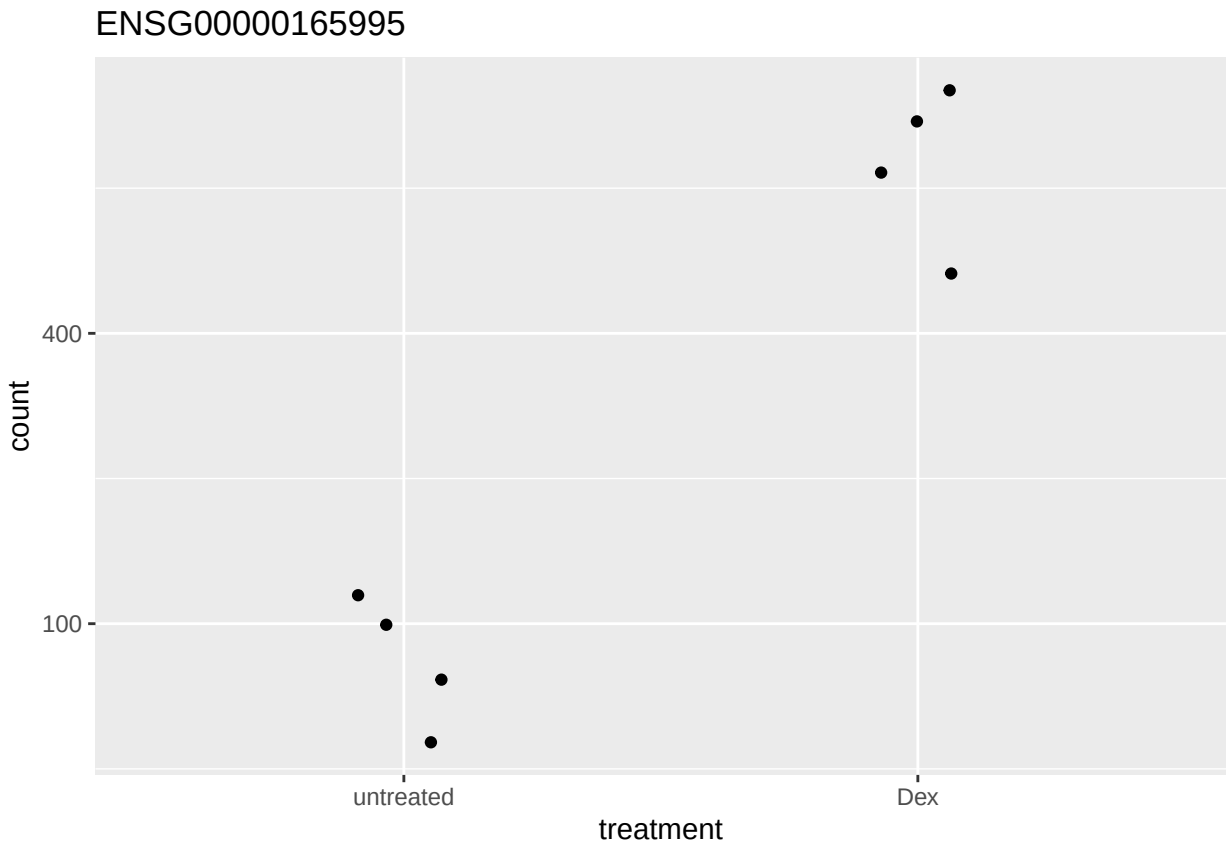



```
plotCounts(dds, gene="ENSG00000000938", intgroup="treatment", main="ENSG00000000938")
```




```
# nicer version using ggplot2
#library("ggplot2")
d <- plotCounts(dds,
                gene=which.min(res$padj),
                intgroup="treatment",
                main=onegene,
```

```
                returnData=TRUE)

ggplot(d, aes(x=treatment, y=count)) +
  geom_point(position=position_jitter(w=0.1,h=0)) +
  scale_y_log10(breaks=c(25,100,400)) +
  labs(title=onegene)
```



Result Information

More information on results columns Information about which variables and tests were used can be found by calling the function mcols on the results object.

```
mcols(res)$description
```

```
## [1] "mean of normalized counts for all samples"
## [2] "log2 fold change (MAP): treatment Dex vs untreated"
## [3] "standard error: treatment Dex vs untreated"
## [4] "Wald statistic: treatment Dex vs untreated"
## [5] "Wald test p-value: treatment Dex vs untreated"
## [6] "BH adjusted p-values"
```

```
# [1] "mean of normalized counts for all samples"
# [2] "log2 fold change (MAP): treatment Dex vs untreated"
# [3] "standard error: treatment Dex vs untreated"
# [4] "Wald statistic: treatment Dex vs untreated"
# [5] "Wald test p-value: treatment Dex vs untreated"
# [6] "BH adjusted p-values"
```

Exporting results to HTML or CSV files

```
write.csv(as.data.frame(resOrdered), row.names = TRUE, file="Dex_vs_untreated_results.csv")
```

```
# write in excel format
write.xlsx(as.data.frame(resOrdered), row.names = TRUE, asTable = TRUE, file="Dex_vs_untreated_results.xls

# count with two cutoffs
print("# rows with abs(LFC) >=1")
```

```
## [1] "# rows with abs(LFC) >=1"
```

```
table(abs(resOrdered$log2FoldChange)>=1)
```

```
##
## FALSE  TRUE
## 27928   802
```

```
print("# rows with padj<0.05")
```

```
## [1] "# rows with padj<0.05"
```

```
table(resOrdered$padj<0.001)
```

```
##
## FALSE  TRUE
## 15102  2026
```

```
table( ( abs(resOrdered$log2FoldChange)>=1) & (resOrdered$padj<0.001) )
```

```
##
## FALSE  TRUE
## 28048   677
```

```
# extract significant subset for padj<0.001
resSig <- subset(resOrdered, padj < 0.001)

# preview results
resSig
```

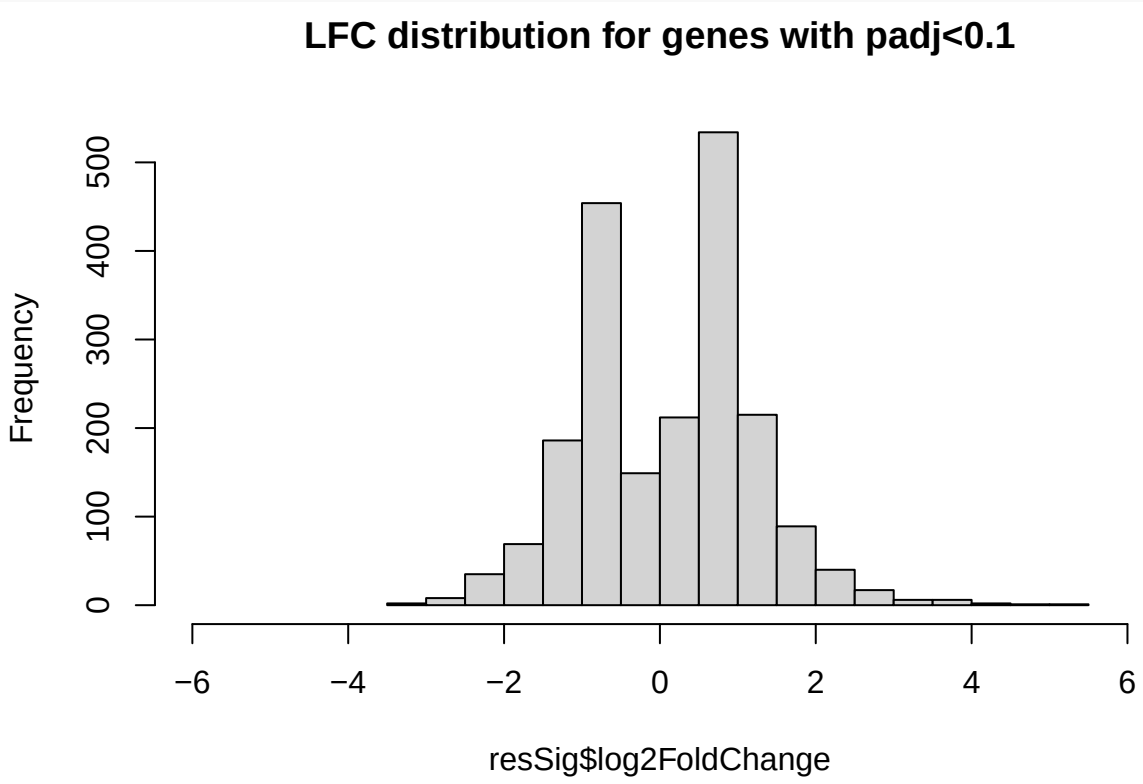
```
## log2 fold change (MAP): treatment Dex vs untreated
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 2026 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat       pvalue
##                  <numeric>      <numeric> <numeric> <numeric>    <numeric>
## ENSG00000165995   514.284        3.32076  0.130973   25.3546 8.00568e-142
## ENSG00000120129  3325.403        2.87348  0.116269   24.7139 7.57320e-135
## ENSG00000152583   985.559        4.34018  0.175930   24.6700 2.24572e-134
## ENSG00000101347 13616.935        3.60737  0.151117   23.8713 6.08347e-126
## ENSG00000189221  2294.730        3.23216  0.139179   23.2231 2.65891e-119
## ...                   ...            ...       ...       ...          ...
## ENSG00000175348  1272.292       0.352301 0.0914507   3.85236  0.000116984
## ENSG00000205302  1598.486      -0.353701 0.0918306  -3.85166  0.000117318
## ENSG00000185344   423.564       0.463839 0.1204356   3.85135  0.000117470
## ENSG00000138744   197.525       0.645667 0.1676964   3.85022  0.000118014
## ENSG00000164342   179.857      -0.635161 0.1649812  -3.84990  0.000118167
##                         padj
##                    <numeric>
## ENSG00000165995 1.37121e-137
## ENSG00000120129 6.48569e-131
## ENSG00000152583 1.28216e-130
## ENSG00000101347 2.60494e-122
## ENSG00000189221 9.10835e-116
## ...                      ...
## ENSG00000175348  0.000990951
## ENSG00000205302  0.000993286
```

```
## ENSG00000185344  0.000994080
## ENSG00000138744  0.000998190
## ENSG00000164342  0.000998993
```

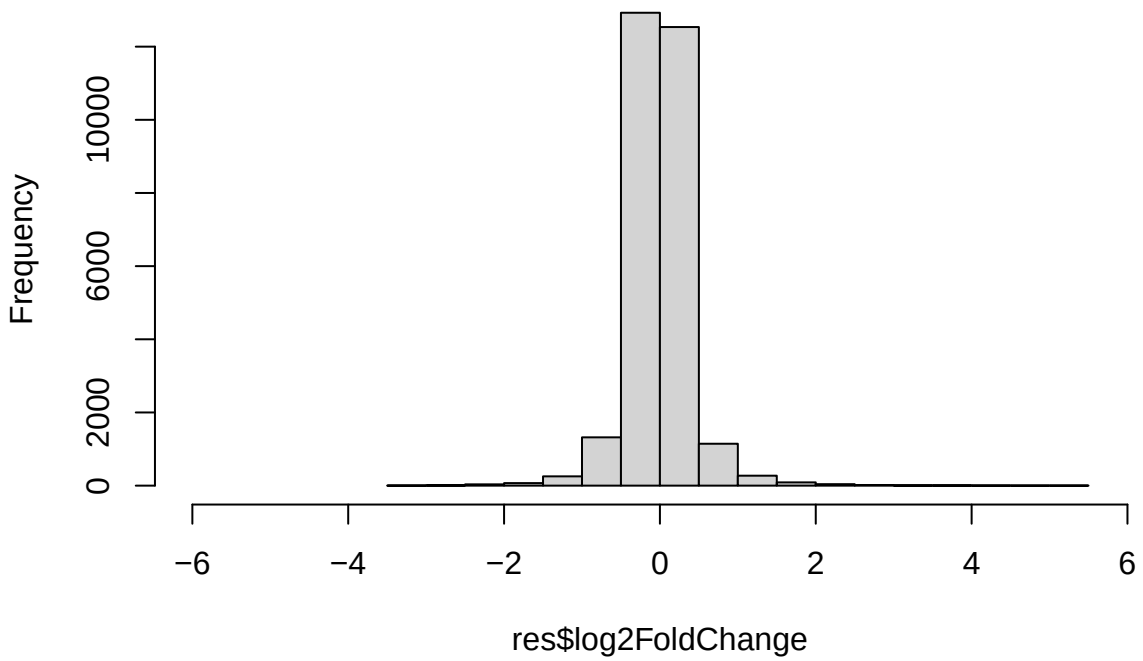
```
# get dimentions
dim(resSig)
```

```
## [1] 2026    6
```

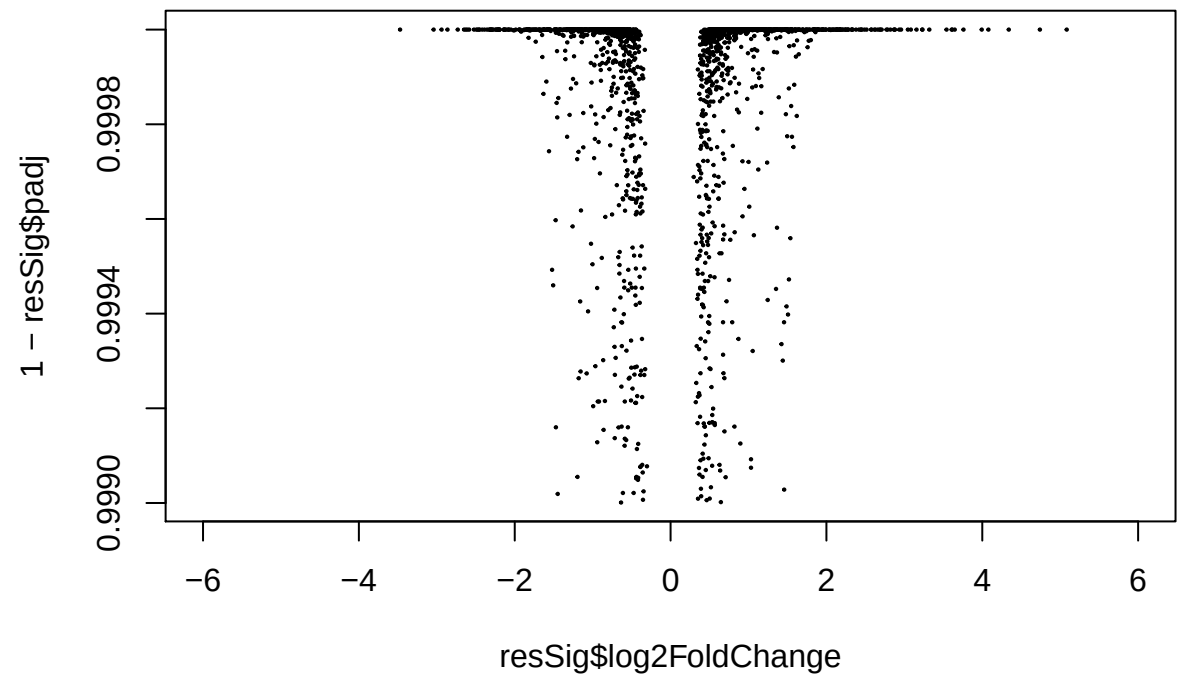
```
# plot distribution of LFC in this subset
hist(resSig$log2FoldChange,
     xlim=c(-6,6),
     breaks=20,
     main="LFC distribution for genes with padj<0.1")
```




```
hist(res$log2FoldChange,
     xlim=c(-6,6),
     breaks=20,
     main="LFC distribution for all genes")
```

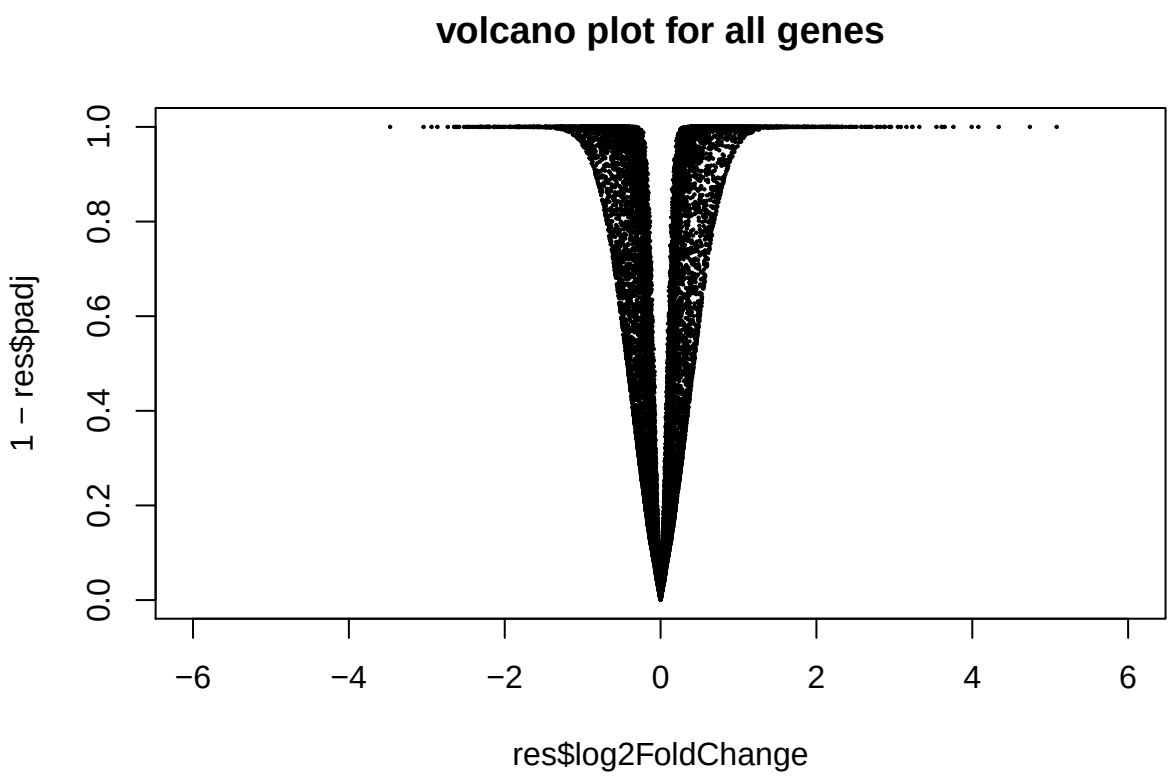



```
# volcano for the subset (using small dots)
plot(resSig$log2FoldChange, 1-resSig$padj,
     xlim=c(-6,6),
     main="volcano plot for genes with padj<0.001",
     pch=20,
     cex=0.25)
```




```
plot(res$log2FoldChange, 1-res$padj,
     xlim=c(-6,6),
     main="volcano plot for all genes",
```

```
     pch=20,
     cex=0.25)
```




```
# find the most affected genes
print("# rows with abs(LFC) >=2 AND padj<0.001")
```

```
## [1] "# rows with abs(LFC) >=2 AND padj<0.001"
```

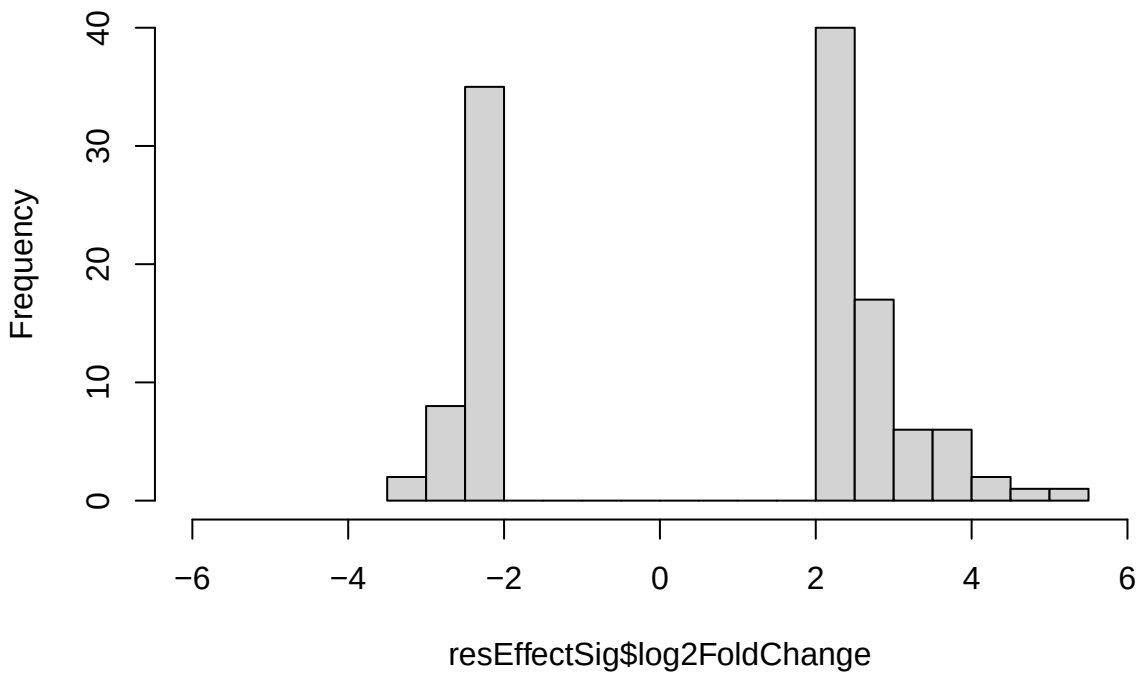
```
table( (abs(resOrdered$log2FoldChange)>=2 & resOrdered$padj<0.001) )
```

```
##
## FALSE  TRUE
## 28612   118
```

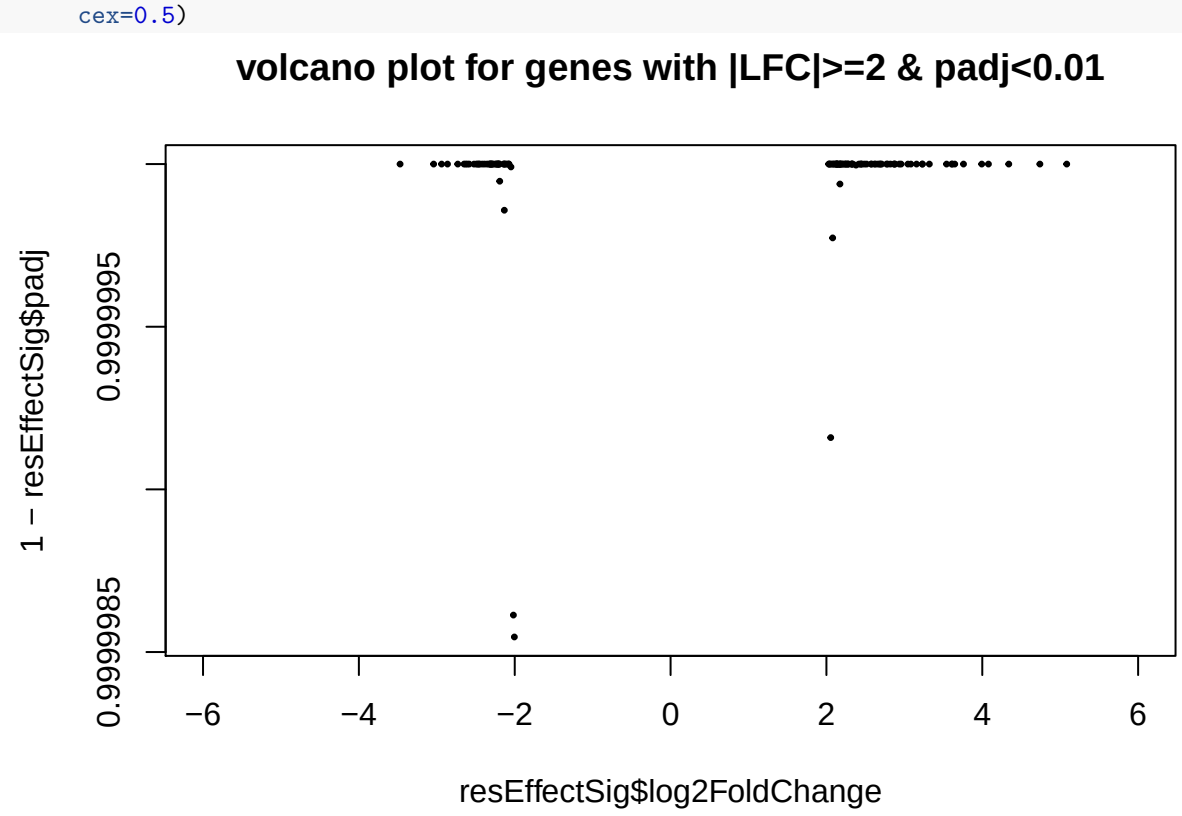
```
# make this a new table
resEffectSig <- subset(resOrdered, ( abs(resOrdered$log2FoldChange)>=2 & padj < 0.001 ))
dim(resEffectSig)
```

```
## [1] 118   6
```

```
# plot distribution of LFC in this subset
hist(resEffectSig$log2FoldChange,
     xlim=c(-6,6),
     breaks=20,
     main="LFC distribution for genes with |LFC|>=2 & padj<0.001")
```




```
# volcano for the subset
plot(resEffectSig$log2FoldChange, 1-resEffectSig$padj,
     xlim=c(-6,6),
     main="volcano plot for genes with |LFC|>=2 & padj<0.01",
     pch=20,
     cex=0.5)
```



Extracting various values

```
# extracting average values
meanval <- assays(dds)[["mu"]]
head(meanval)
```

```
##                  SRR1039508  SRR1039509  SRR1039512  SRR1039513   SRR1039516
## ENSG00000000003 656.6968708 441.4402201 888.6713765 387.6041150 1143.0590567
## ENSG00000000005          NA          NA          NA          NA           NA
## ENSG00000000419 452.5563157 464.9319872 529.5120015 352.9659719  522.7172321
## ENSG00000000457 254.5589602 227.3201139 287.3436204 166.4911949  292.0572569
## ENSG00000000460  57.7281707  50.4463898  39.8569674  22.5988730   53.7372175
## ENSG00000000938   0.2883364   0.1173442   0.8463658   0.2234912    0.6233371
##                  SRR1039517 SRR1039520  SRR1039521
## ENSG00000000003 1040.7572372 732.003714 576.7080563
## ENSG00000000005           NA         NA          NA
## ENSG00000000419  727.3734282 383.429782 461.6771923
## ENSG00000000457  353.2574228 236.206774 247.2165867
## ENSG00000000460   63.6050046  61.323869  62.8069384
## ENSG00000000938    0.3436046   0.266337   0.1270368
```

```
# extractiong Cook distance values
cookdist <- assays(dds)[["cooks"]]
head(cookdist)
```

```
##                   SRR1039508  SRR1039509 SRR1039512 SRR1039513  SRR1039516
## ENSG00000000003 0.0054983546 0.005172100 0.02237698 0.01984628 0.001764613
## ENSG00000000005           NA          NA         NA         NA          NA
## ENSG00000000419 0.0518558931 0.052066233 0.05749710 0.05384144 0.002999874
## ENSG00000000457 0.0006661005 0.000648921 0.03682720 0.03208024 0.003954933
## ENSG00000000460 0.0104078512 0.009929868 0.84967836 0.62138610 0.318277883
## ENSG00000000938 0.2505599108 0.102664653 0.02680625 0.18329431 0.206883747
##                  SRR1039517  SRR1039520  SRR1039521
## ENSG00000000003 0.001751567 0.013309565 0.012906464
## ENSG00000000005          NA          NA          NA
## ENSG00000000419 0.003116652 0.003941036 0.004063574
## ENSG00000000457 0.004090077 0.049163935 0.049676474
## ENSG00000000460 0.336284062 0.137046249 0.138065710
## ENSG00000000938 0.289416808 0.231644279 0.111101822
```

Extracting transformed values

Transformed values are better for representation like heatmaps as they show a more normal distribution of color intensities.

```
# Regularized log transformation
rld <- rlog(dds)
rld
```

```
## class: DESeqTransform
## dim: 57773 8
## metadata(1): version
## assays(1): ''
## rownames(57773): ENSG00000000003 ENSG00000000005 ... ENSG00000273492
##   ENSG00000273493
## rowData names(48): baseMean baseVar ... dispFit rlogIntercept
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(3): cells treatment sizeFactor
```

```
# Variance stabilizing transformation
vsd <- varianceStabilizingTransformation(dds)
vsd
```

```
## class: DESeqTransform
## dim: 57773 8
## metadata(1): version
## assays(1): ''
## rownames(57773): ENSG00000000003 ENSG00000000005 ... ENSG00000273492
##   ENSG00000273493
## rowData names(47): baseMean baseVar ... dispGeneIter dispFit
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(3): cells treatment sizeFactor
```

```
# create matrix for saving or further use
rlogMat <- assay(rld)
head(rlogMat)
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003   9.385816   9.111932   9.485294   9.293744   9.745661
## ENSG00000000005   0.000000   0.000000   0.000000   0.000000   0.000000
## ENSG00000000419   8.790052   9.016457   8.887313   8.928874   8.843570
## ENSG00000000457   7.979714   7.978258   7.915775   7.997562   7.953958
## ENSG00000000460   5.702620   5.735181   5.226853   5.562929   5.683211
## ENSG00000000938  -1.732975  -1.731689  -1.713623  -1.728722  -1.713608
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003   9.502456   9.584328   9.285420
## ENSG00000000005   0.000000   0.000000   0.000000
## ENSG00000000419   8.958819   8.761472   8.921047
## ENSG00000000457   7.986458   8.036349   7.958016
## ENSG00000000460   5.454298   5.909072   5.743929
## ENSG00000000938  -1.736400  -1.731967  -1.732341
```

```
# create matrix for saving or further use
vstMat <- assay(vsd)
head(vstMat)
```

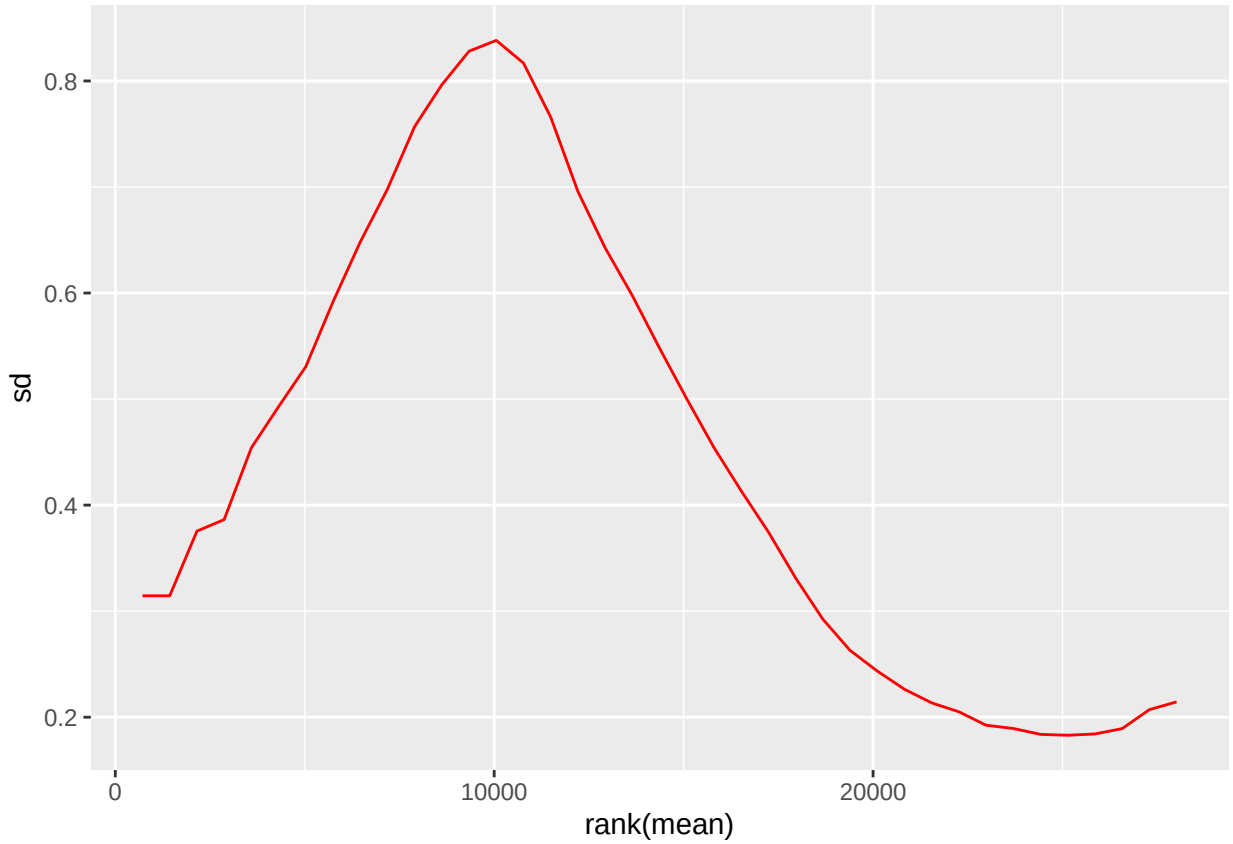
```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003   9.434561   9.016186   9.582980   9.294669   9.964544
## ENSG00000000005   4.066499   4.066499   4.066499   4.066499   4.066499
## ENSG00000000419   8.838104   9.175313   8.983751   9.045842   8.918612
## ENSG00000000457   8.156858   8.154731   8.062923   8.183414   8.119083
## ENSG00000000460   6.418609   6.466341   5.724891   6.214102   6.390111
## ENSG00000000938   4.066499   4.066499   4.391008   4.066499   4.391245
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003   9.608287   9.729634   9.282903
## ENSG00000000005   4.066499   4.066499   4.066499
## ENSG00000000419   9.089485   8.794595   9.033904
## ENSG00000000457   8.166662   8.240069   8.124917
## ENSG00000000460   6.067098   6.711486   6.477942
## ENSG00000000938   4.066499   4.066499   4.066499
```

Effects of transformations on the variance

```
# library("vsn")
par(mfrow=c(1,3))
notAllZero <- (rowSums(counts(dds))>0)
meanSdPlot(log2(counts(dds,normalized=TRUE)[notAllZero,] + 1), main="log2 tranformation")
```

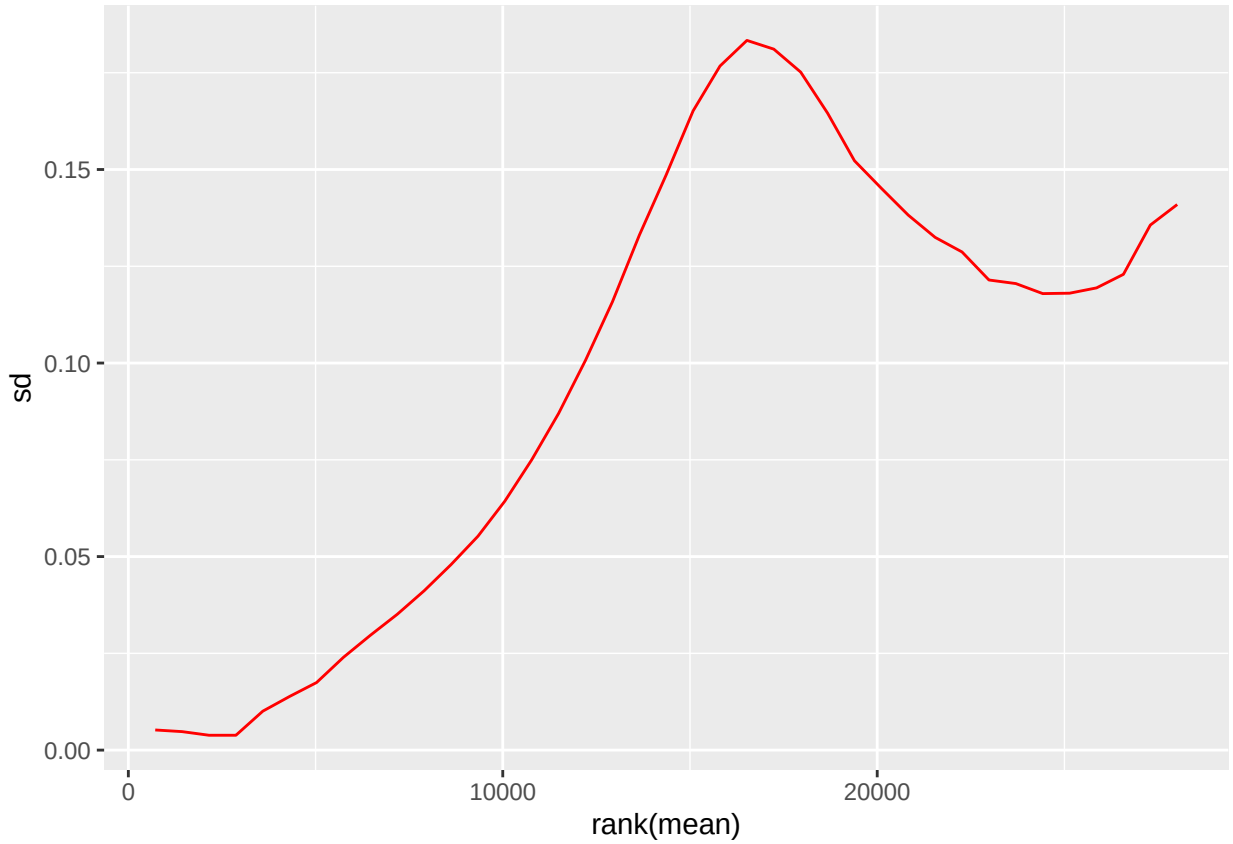
```
## Warning: Ignoring unknown parameters: main
```

```
## Warning: Computation failed in `stat_binhex()`:
##   Package `hexbin` required for `stat_binhex`.
##   Please install and try again.
```




```
meanSdPlot(assay(rld[notAllZero,]), main="Regularized log transformation")
```

```
## Warning: Ignoring unknown parameters: main

## Warning: Computation failed in `stat_binhex()`:
##   Package `hexbin` required for `stat_binhex`.
##   Please install and try again.
```




```
meanSdPlot(assay(vsd[notAllZero,]), main="Variance stabilizing transformation")
```

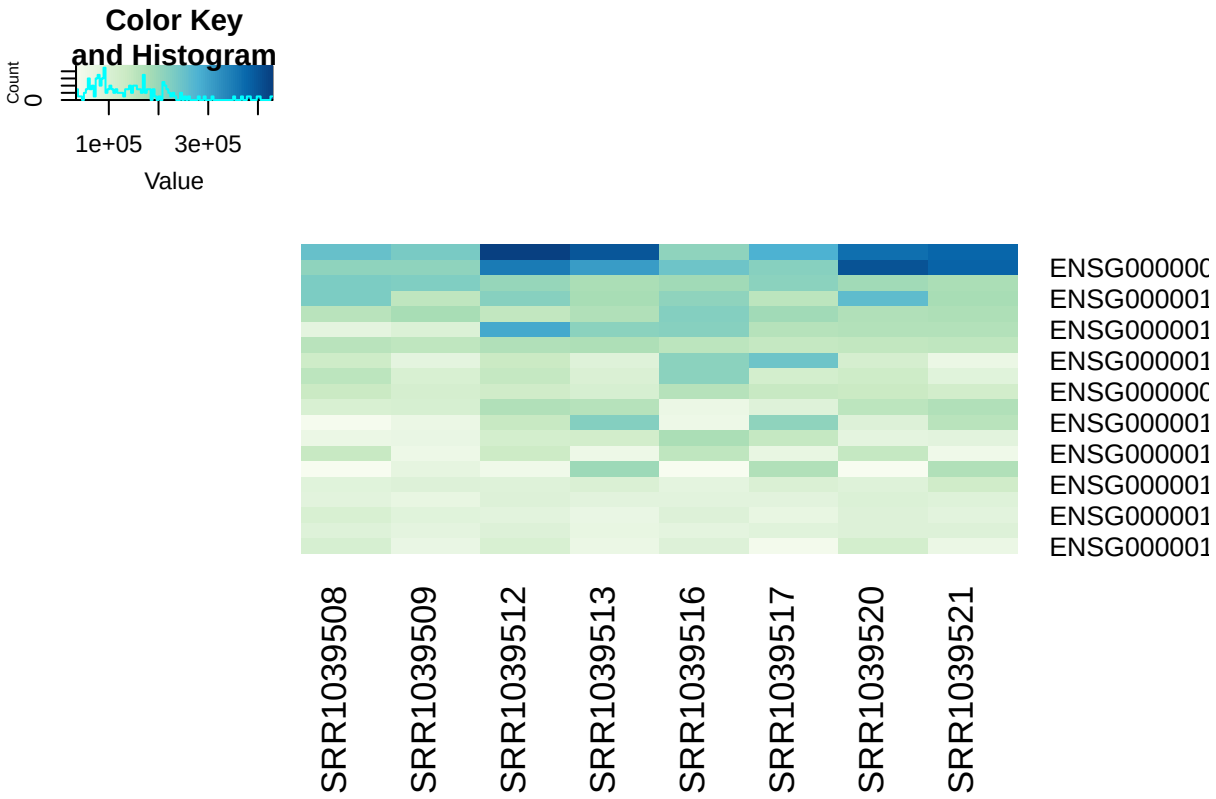
```
## Warning: Ignoring unknown parameters: main

## Warning: Computation failed in `stat_binhex()`:
##   Package `hexbin` required for `stat_binhex`.
##   Please install and try again.
```


```
# legend: Standard deviation over mean. Per-gene standard deviation (taken across samples), against the ra
```

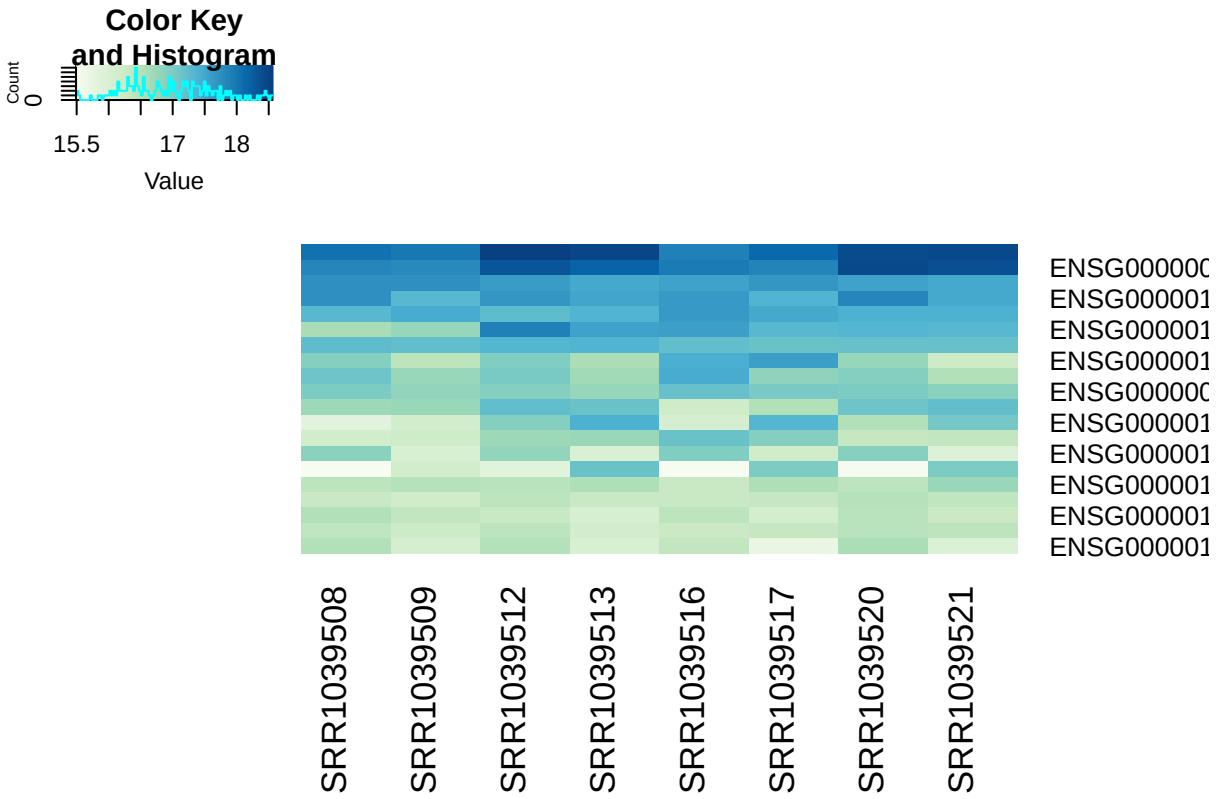
Data quality assessment by sample clustering and visualization

Heatmap of the count matrix

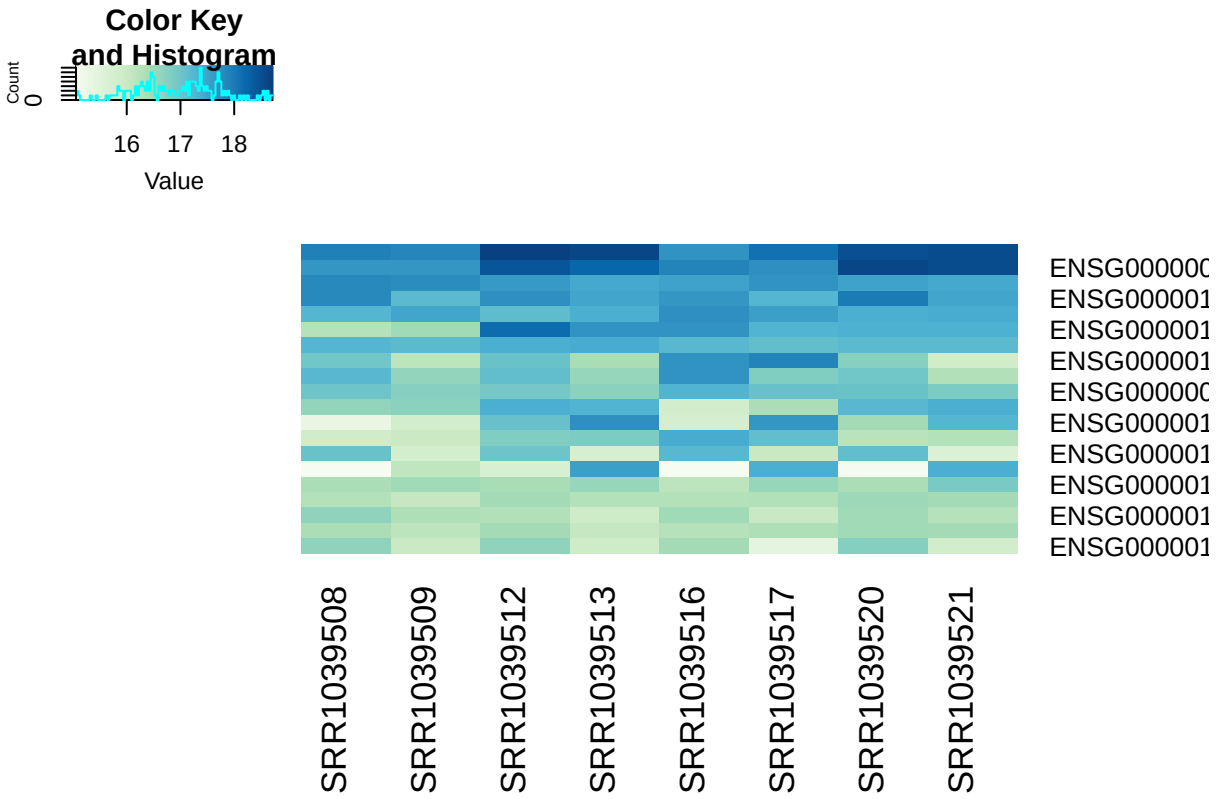
```
#library("RColorBrewer")
#library("gplots")

# color palette
hmcol <- colorRampPalette(brewer.pal(9, "GnBu"))(100)

# select top 20 genes
n=20
select <- order(rowMeans(counts(dds,normalized=TRUE)), decreasing=TRUE)[1:n]

# show from counts
heatmap.2(counts(dds,normalized=TRUE)[select,],
          col = hmcol,
          Rowv = FALSE,
          Colv = FALSE,
          scale="none",
          dendrogram="none",
          trace="none",
          margin=c(10,6)
          )
```



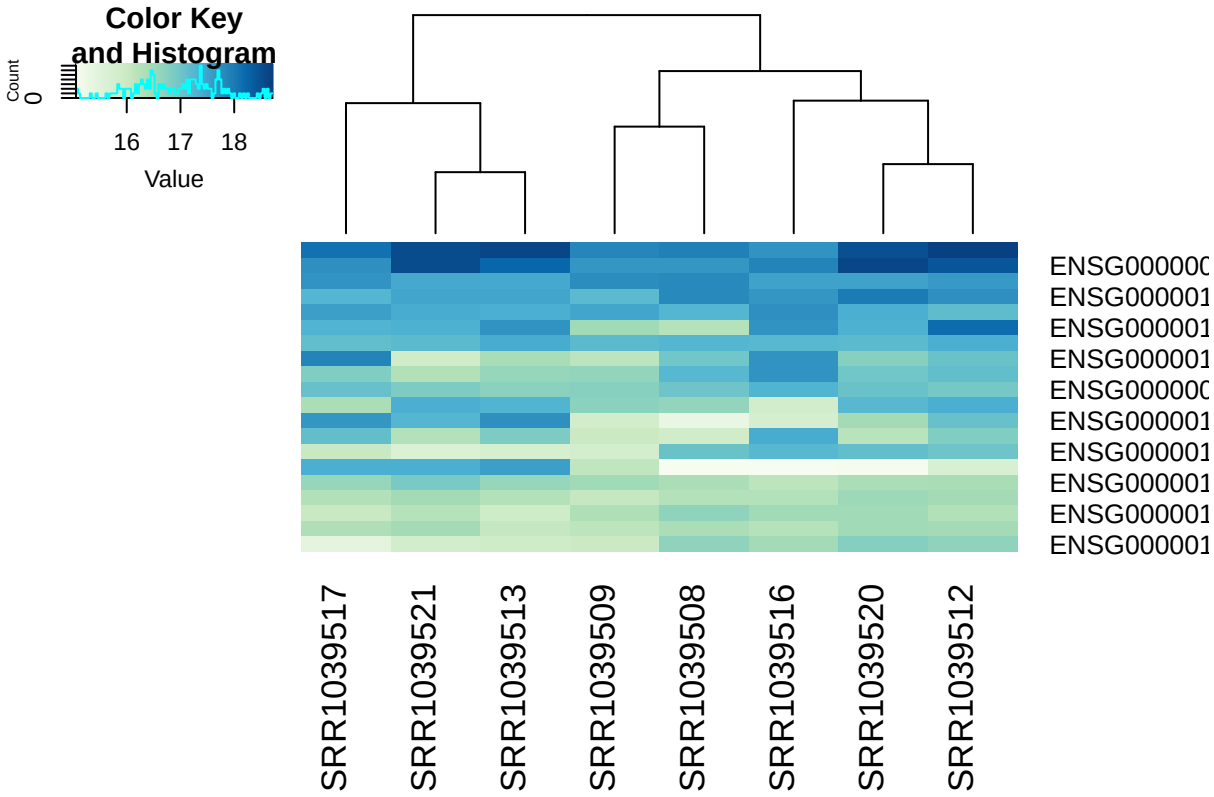

```
# show from Regularized log transformation
heatmap.2(assay(rld)[select,],
          col = hmcol,
          Rowv = FALSE,
          Colv = FALSE,
          scale="none",
          dendrogram="none",
          trace="none",
          margin=c(10, 6)
          )
```




```
# show from Variance stabilizing transformation
heatmap.2(assay(vsd)[select,],
          col = hmcol,
          Rowv = FALSE,
          Colv = FALSE,
          scale="none",
          dendrogram="none",
          trace="none",
          margin=c(10, 6)
          )
```

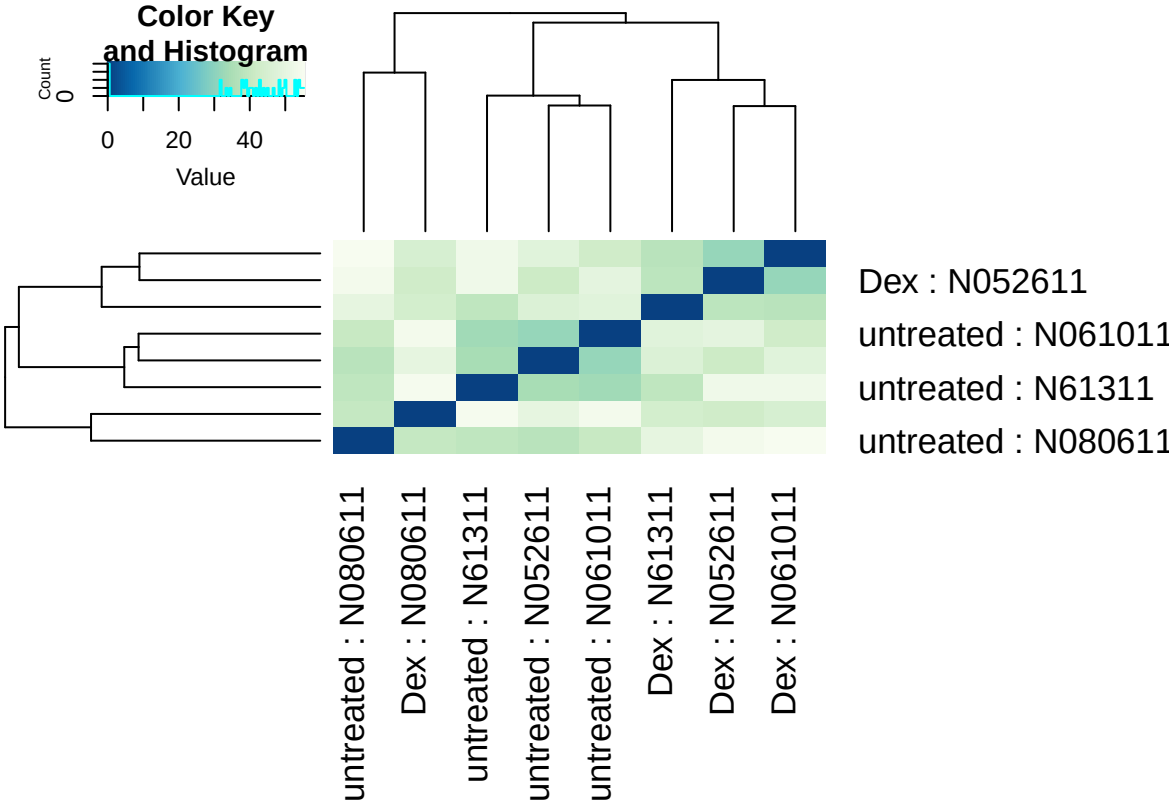



```
# show from Variance stabilizing transformation with sample clustering
heatmap.2(assay(vsd)[select,],
          col = hmcol,
          Rowv = FALSE,
          Colv = TRUE,
          scale="none",
          dendrogram="column",
          trace="none",
          margin=c(10, 6)
          )
```



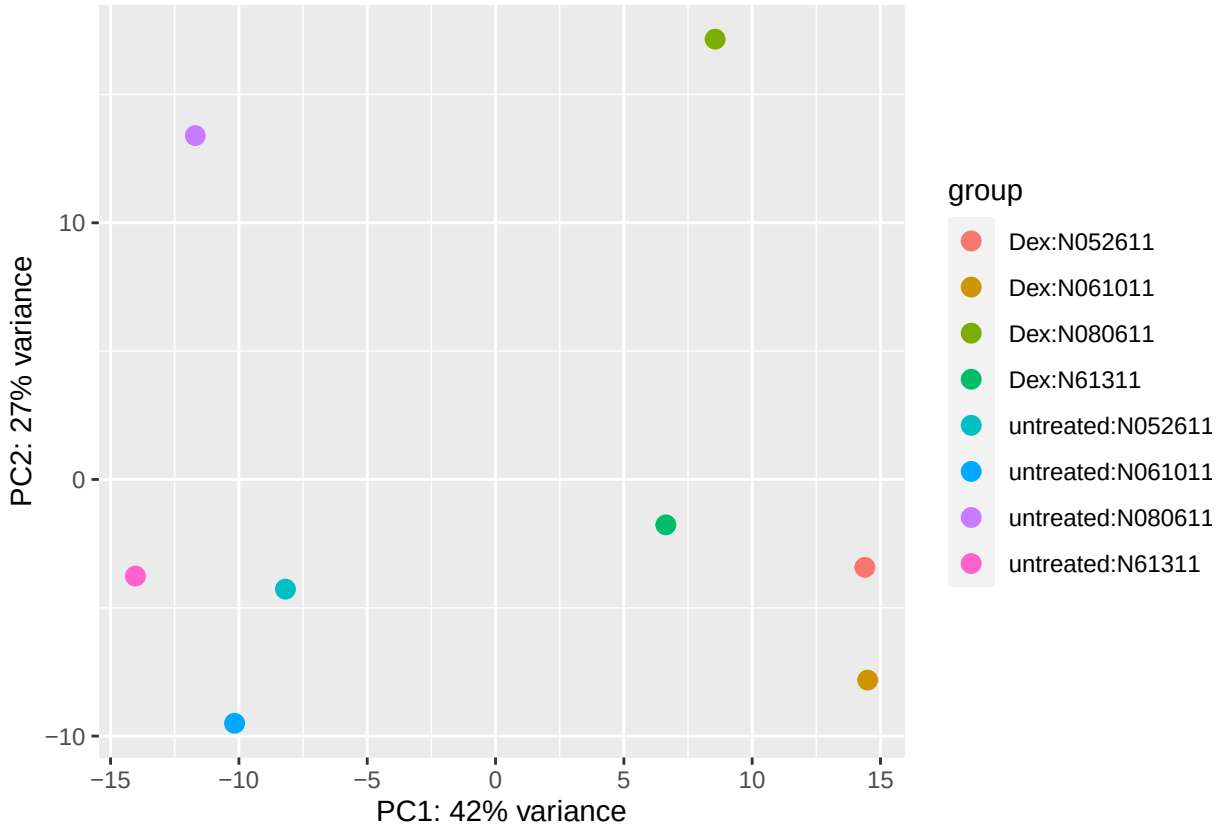
Heatmap of the sample-to-sample distances

```
# we need to transpose the data first with t()
distsRL <- dist(t(assay(rld)))

# store in to matrix
mat <- as.matrix(distsRL)

# rename samples
rownames(mat) <- colnames(mat) <- with(colData(dds),
          paste(treatment, cells, sep=" : "))

# cluster pairwise
hc <- hclust(distsRL)

# plot heatmap
heatmap.2(mat,
          Rowv=as.dendrogram(hc),
          symm=TRUE,
          trace="none",
          col = rev(hmcol),
          margin=c(13, 13)
          )
```

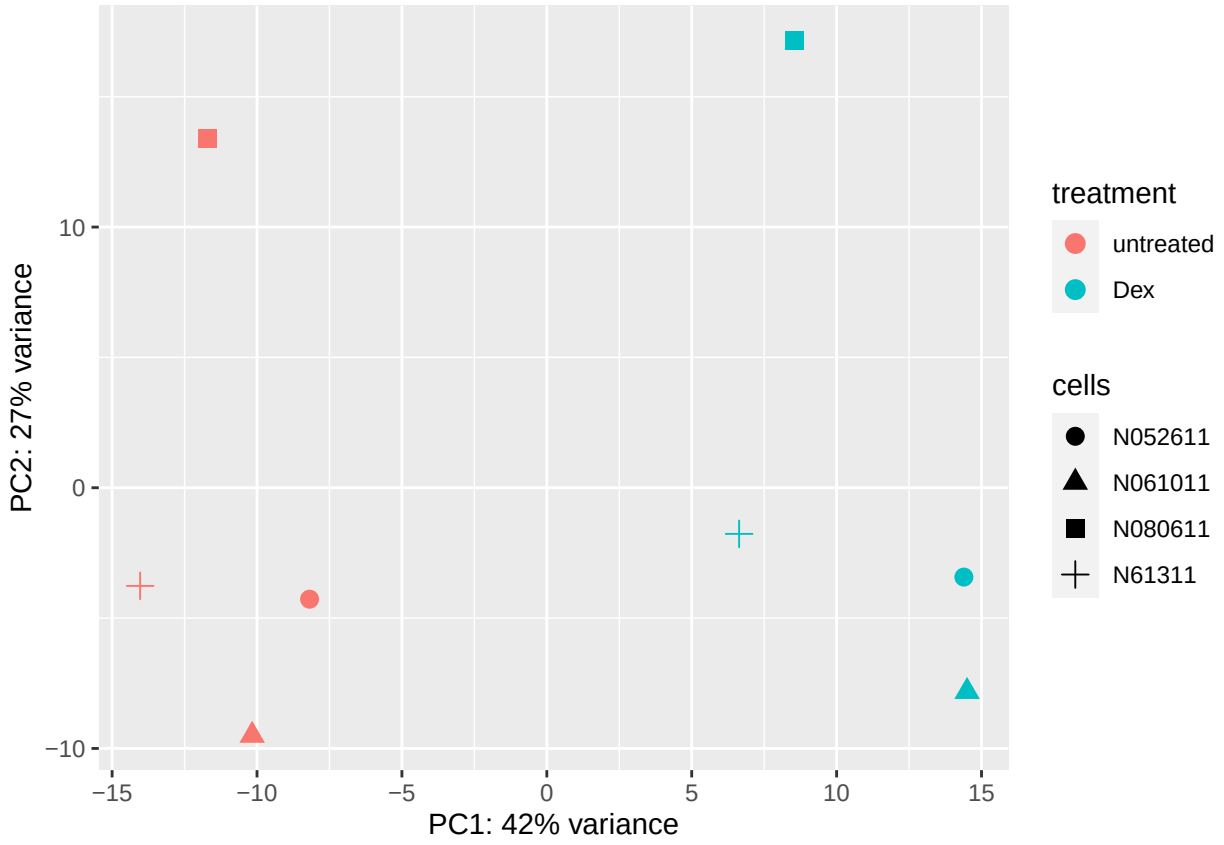


Principal component plot of the samples

```
# simple plot without grouping
plotPCA(rld, intgroup=c("treatment", "cells"))
```



```
# It is also possible to customize the PCA plot using the ggplot function.
data <- plotPCA(rld, intgroup=c("treatment", "cells"), returnData=TRUE)
percentVar <- round(100 * attr(data, "percentVar"))

ggplot(data, aes(PC1, PC2, color=treatment, shape=cells)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance"))
```



Tests of log2 fold change above or below a threshold

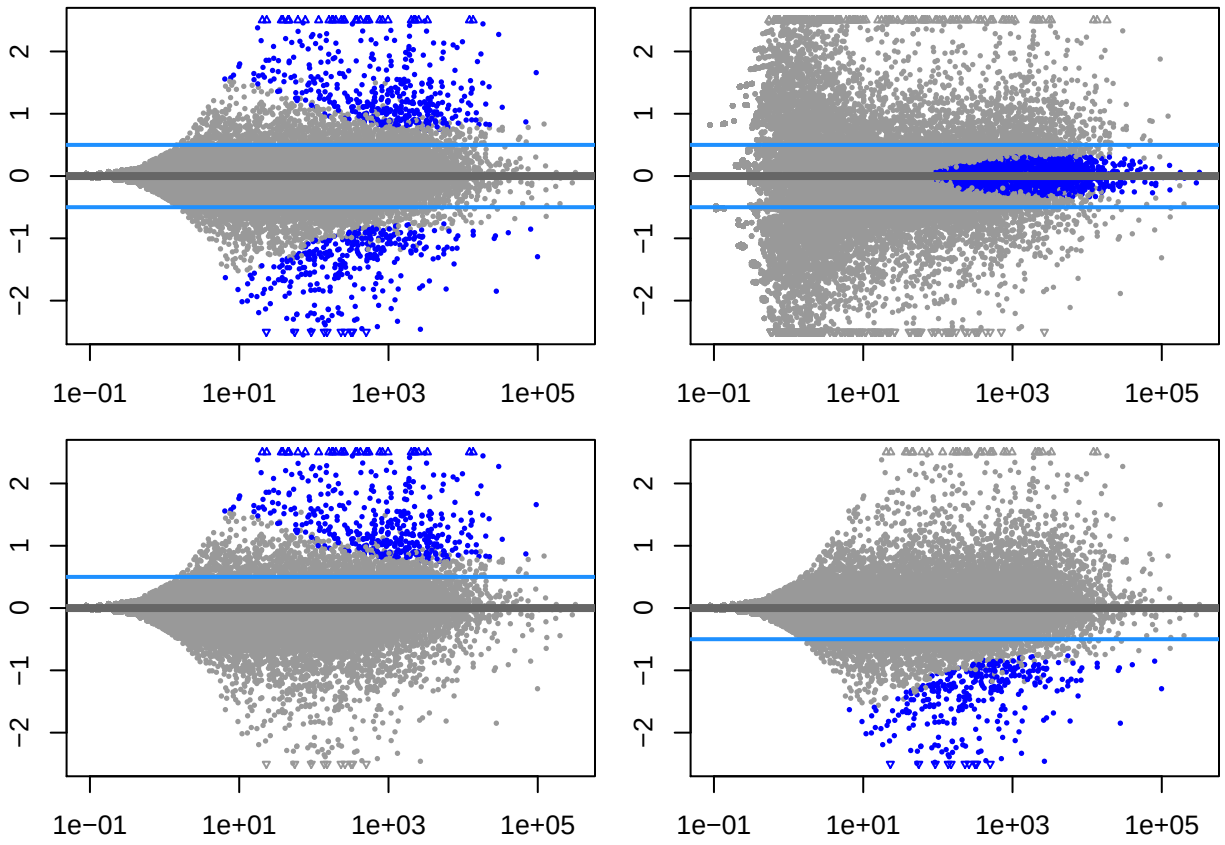
Other ways of testing by choosing the minimal effect size and direction (tails)

- greaterAbs - |beta| > lfcThreshold - tests are two-tailed
- lessAbs - |beta| < lfcThreshold - p values are the maximum of the upper and lower tests
- greater - beta > lfcThreshold
- less - beta < lfcThreshold

```
# create a new object without priors
ddsNoPrior <- DESeq(dds, betaPrior=FALSE)
```

```
## using pre-existing size factors

## estimating dispersions

## found already estimated dispersions, replacing these

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
# compute four tests
resGA <- results(dds, lfcThreshold=.5, altHypothesis="greaterAbs")
resLA <- results(ddsNoPrior, lfcThreshold=.5, altHypothesis="lessAbs")
resG <- results(dds, lfcThreshold=.5, altHypothesis="greater")
resL <- results(dds, lfcThreshold=.5, altHypothesis="less")

# plot MA for each test
par(mfrow=c(2,2),mar=c(2,2,1,1))
yl <- c(-2.5,2.5)

{
  plotMA(resGA, ylim=yl)
abline(h=c(-.5,.5),col="dodgerblue",lwd=2)
}

{
plotMA(resLA, ylim=yl)
abline(h=c(-.5,.5),col="dodgerblue",lwd=2)
}

{
plotMA(resG, ylim=yl)
abline(h=.5,col="dodgerblue",lwd=2)
}

{
plotMA(resL, ylim=yl)
abline(h=-.5,col="dodgerblue",lwd=2)
}
```



```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] openxlsx_4.1.5              gplots_3.0.3
##  [3] RColorBrewer_1.1-2          vsn_3.56.0
##  [5] ggplot2_3.3.1               DESeq2_1.28.1
##  [7] SummarizedExperiment_1.18.1 DelayedArray_0.14.0
##  [9] matrixStats_0.56.0          Biobase_2.48.0
## [11] GenomicRanges_1.40.0        GenomeInfoDb_1.24.0
## [13] IRanges_2.22.2              S4Vectors_0.26.1
## [15] BiocGenerics_0.34.0
##
## loaded via a namespace (and not attached):
##  [1] bit64_0.9-7            splines_4.0.0          gtools_3.8.2
##  [4] BiocManager_1.30.10    affy_1.66.0            blob_1.2.1
##  [7] GenomeInfoDbData_1.2.3 yaml_2.2.1             pillar_1.4.4
## [10] RSQLite_2.2.0          lattice_0.20-41        glue_1.4.1
## [13] limma_3.44.3           digest_0.6.25          XVector_0.28.0
## [16] colorspace_1.4-1       htmltools_0.4.0        preprocessCore_1.50.0
## [19] Matrix_1.2-18          XML_3.99-0.3           pkgconfig_2.0.3
## [22] genefilter_1.70.0      zlibbioc_1.34.0        purrr_0.3.4
## [25] xtable_1.8-4           scales_1.1.1           gdata_2.18.0
## [28] affyio_1.58.0          BiocParallel_1.22.0    tibble_3.0.1
## [31] annotate_1.66.0        farver_2.0.3           generics_0.0.2
## [34] ellipsis_0.3.1         withr_2.2.0            survival_3.2-3
## [37] magrittr_1.5           crayon_1.3.4           memoise_1.1.0
## [40] evaluate_0.14          tools_4.0.0            lifecycle_0.2.0
## [43] stringr_1.4.0          munsell_0.5.0          locfit_1.5-9.4
## [46] zip_2.0.4              AnnotationDbi_1.50.0   compiler_4.0.0
## [49] caTools_1.18.0         rlang_0.4.6            grid_4.0.0
## [52] RCurl_1.98-1.2         labeling_0.3           bitops_1.0-6
## [55] rmarkdown_2.2          gtable_0.3.0           DBI_1.1.0
## [58] R6_2.4.1               knitr_1.28             dplyr_1.0.0
## [61] bit_1.1-15.2           KernSmooth_2.23-17     stringi_1.4.6
## [64] Rcpp_1.0.4.6           vctrs_0.3.1            geneplotter_1.66.0
## [67] tidyselect_1.1.0       xfun_0.14
```