DESeq2_analysis of the SRP033351 data

SP:BITS

April 27, 2015

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")
```

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"
setwd(basedir)


# load metadata
metadata <- read.table("/media/bits/RNASeq_DATA/input_files/GSE52778_metadata.txt", 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 <- "/media/bits/RNASeq_DATA/htseq_counts"

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

```
## factor levels were dropped which had no samples
```

```
# review the object
dds
```

```
## class: DESeqDataSet
## dim: 57773 8
## exptData(0):
## assays(1): counts
## rownames(57773): ENSG00000000003 ENSG00000000005 ...
##   ENSG00000273492 ENSG00000273493
## rowData metadata column 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

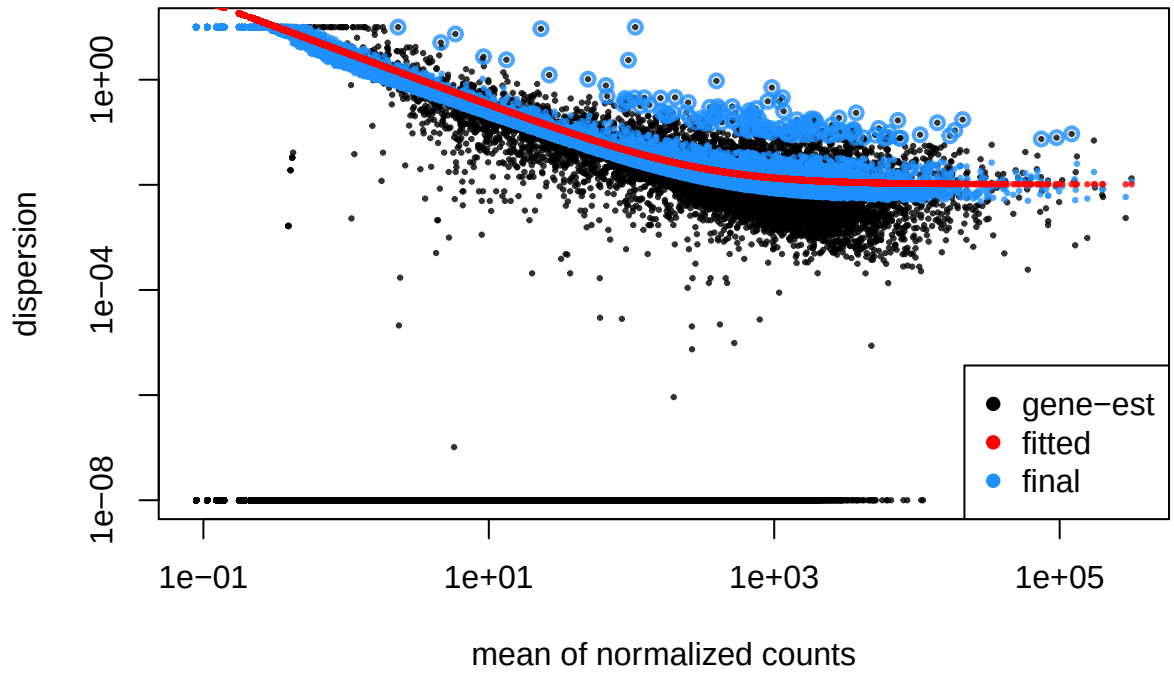
The DESeq function is a wrapper that performs a default analysis through three steps:

- 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)
```

```
## 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
##  1.0123484  0.8951975  1.1747041  0.6739964  1.1729844  1.4049307
## SRR1039520 SRR1039521
##  0.9193101  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
##                   <numeric>      <numeric> <numeric>  <numeric>
## ENSG00000000003 698.9254902    -0.38812002 0.1001493 -3.8754129
## ENSG00000000005   0.0000000             NA        NA         NA
## ENSG00000000419 475.4853185     0.21128035 0.1131837  1.8667038
## ENSG00000000457 251.9014951     0.01368663 0.1325385  0.1032653
## ENSG00000000460  50.8475599    -0.02705074 0.2608112 -0.1037177
## ENSG00000000938   0.2129756    -0.10524156 0.1416069 -0.7431954
```

```
##                      pvalue         padj
##                   <numeric>    <numeric>
## ENSG00000000003 0.0001064441 0.0008072674
## ENSG00000000005           NA           NA
## ENSG00000000419 0.0619429685 0.1695057179
## ENSG00000000457 0.9177523856 0.9590952695
## ENSG00000000460 0.9173933898 0.9589720871
## ENSG00000000938 0.4573633907           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
##                  <numeric>      <numeric> <numeric> <numeric>
## ENSG00000165995   514.2841       3.321662 0.1307366  25.40728
## ENSG00000152583   985.5593       4.340812 0.1760858  24.65169
## ENSG00000120129  3325.4027       2.873150 0.1167734  24.60448
## ENSG00000101347 13616.9348       3.606558 0.1517550  23.76566
## ENSG00000189221  2294.7300       3.231983 0.1396172  23.14888
## ENSG00000211445 12162.4869       3.540681 0.1573440  22.50281
##                      pvalue          padj
##                   <numeric>     <numeric>
## ENSG00000165995 2.095444e-142 3.170407e-138
## ENSG00000152583 3.529479e-134 2.670051e-130
## ENSG00000120129 1.131211e-133 5.705072e-130
## ENSG00000101347 7.569341e-125 2.863103e-121
## ENSG00000189221 1.491976e-118 4.514718e-115
## ENSG00000211445 3.895920e-112 9.824213e-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)     : 2504, 8.7%
## LFC < 0 (down)   : 2197, 7.6%
## outliers [1]     : 0, 0%
## low counts [2]   : 13600, 47%
## (mean count < 11.2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

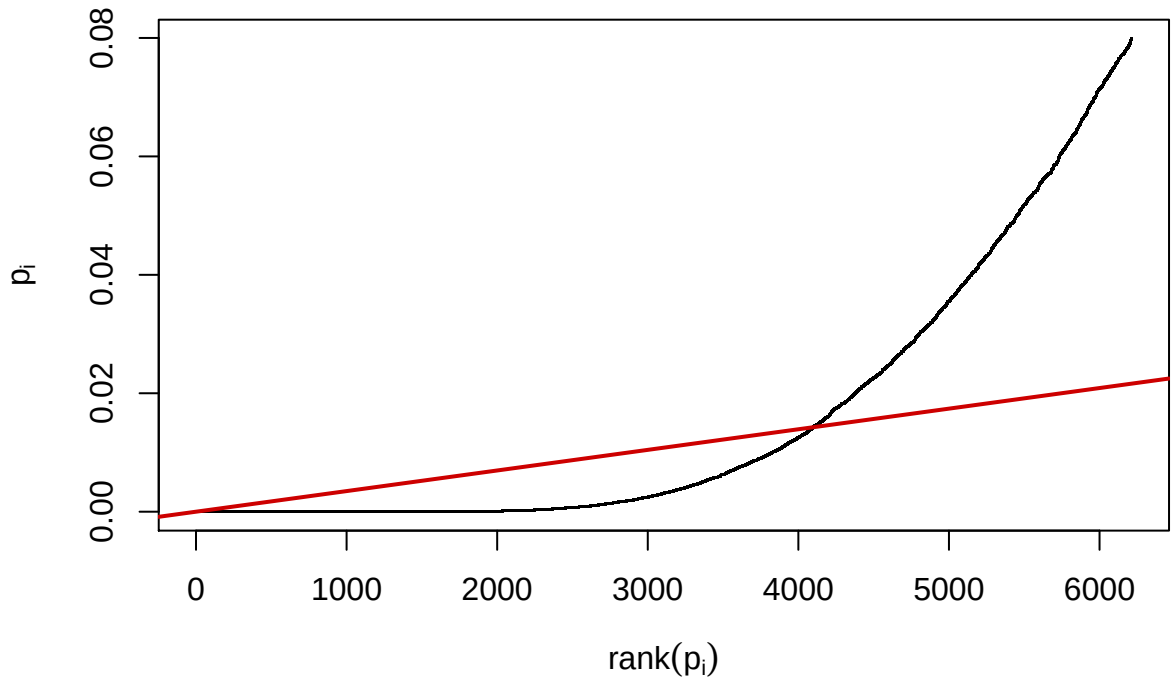
```
## being more stringent
summary(res, alpha=0.01)
```

```
##
## out of 28730 with nonzero total read count
## adjusted p-value < 0.01
## LFC > 0 (up)     : 1562, 5.4%
## LFC < 0 (down)   : 1284, 4.5%
## outliers [1]     : 0, 0%
## low counts [2]   : 13600, 47%
## (mean count < 11.2)
## [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)
```

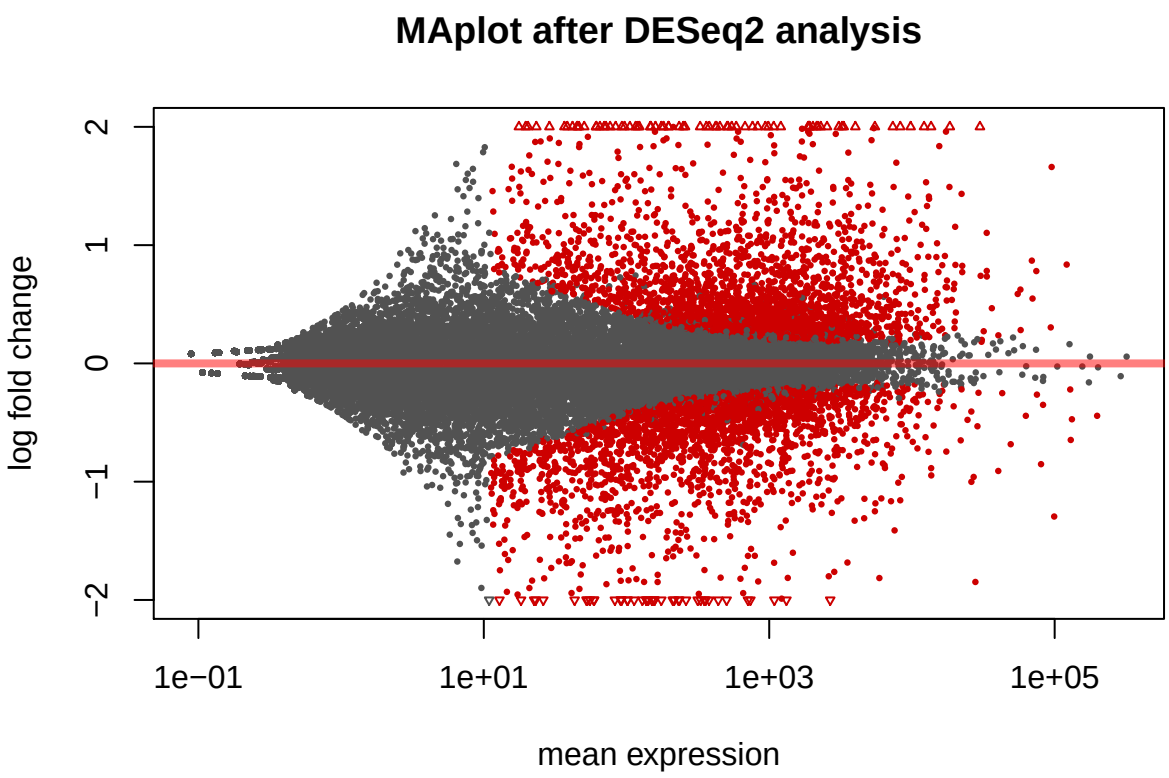
```
## log2 fold change (MAP): treatment Dex vs untreated
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 4 rows and 6 columns
##                  baseMean log2FoldChange     lfcSE       stat       pvalue
##                 <numeric>      <numeric> <numeric>  <numeric>    <numeric>
```

```
## ENSG00000000003  698.9255    -0.38812002 0.1001493 -3.8754129 0.0001064441
## ENSG00000000005    0.0000             NA        NA         NA           NA
## ENSG00000000419  475.4853     0.21128035 0.1131837  1.8667038 0.0619429685
## ENSG00000000457  251.9015     0.01368663 0.1325385  0.1032653 0.9177523856
##                        padj
##                   <numeric>
## ENSG00000000003 0.0008072674
## ENSG00000000005           NA
## ENSG00000000419 0.1695057179
## ENSG00000000457 0.9590952695
```

```
# 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 698.92549    -0.38812002 0.1001493 -3.8754129 0.0001064441
## ENSG00000000419 475.48532     0.21128035 0.1131837  1.8667038 0.0619429685
## ENSG00000000457 251.90150     0.01368663 0.1325385  0.1032653 0.9177523856
## ENSG00000000460  50.84756    -0.02705074 0.2608112 -0.1037177 0.9173933898
##                        padj
##                   <numeric>
## ENSG00000000003 0.0008072674
## ENSG00000000419 0.1695057179
## ENSG00000000457 0.9590952695
## ENSG00000000460 0.9589720871
```

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

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

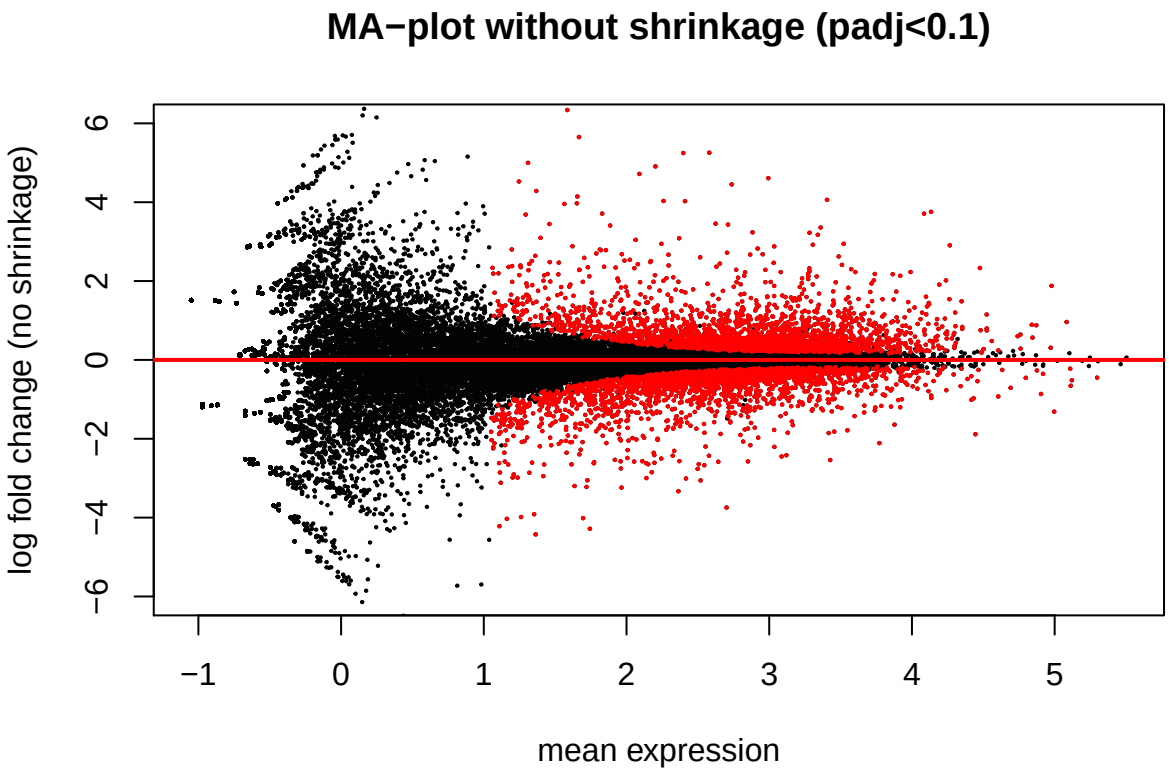



```
# A column lfcMLE with the unshrunken maximum likelihood estimate (MLE) for the log2 fold change will b
resMLE <- results(dds, addMLE=TRUE)
head(resMLE, 4)
```

```
## 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  698.9255    -0.38812002 -0.39558118 0.1001493 -3.8754129
## ENSG00000000005    0.0000             NA          NA        NA         NA
## ENSG00000000419  475.4853     0.21128035  0.21635043 0.1131837  1.8667038
## ENSG00000000457  251.9015     0.01368663  0.01414933 0.1325385  0.1032653
##                       pvalue         padj
##                    <numeric>    <numeric>
## ENSG00000000003 0.0001064441 0.0008072674
## ENSG00000000005           NA           NA
## ENSG00000000419 0.0619429685 0.1695057179
## ENSG00000000457 0.9177523856 0.9590952695
```

```
# 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)")
```

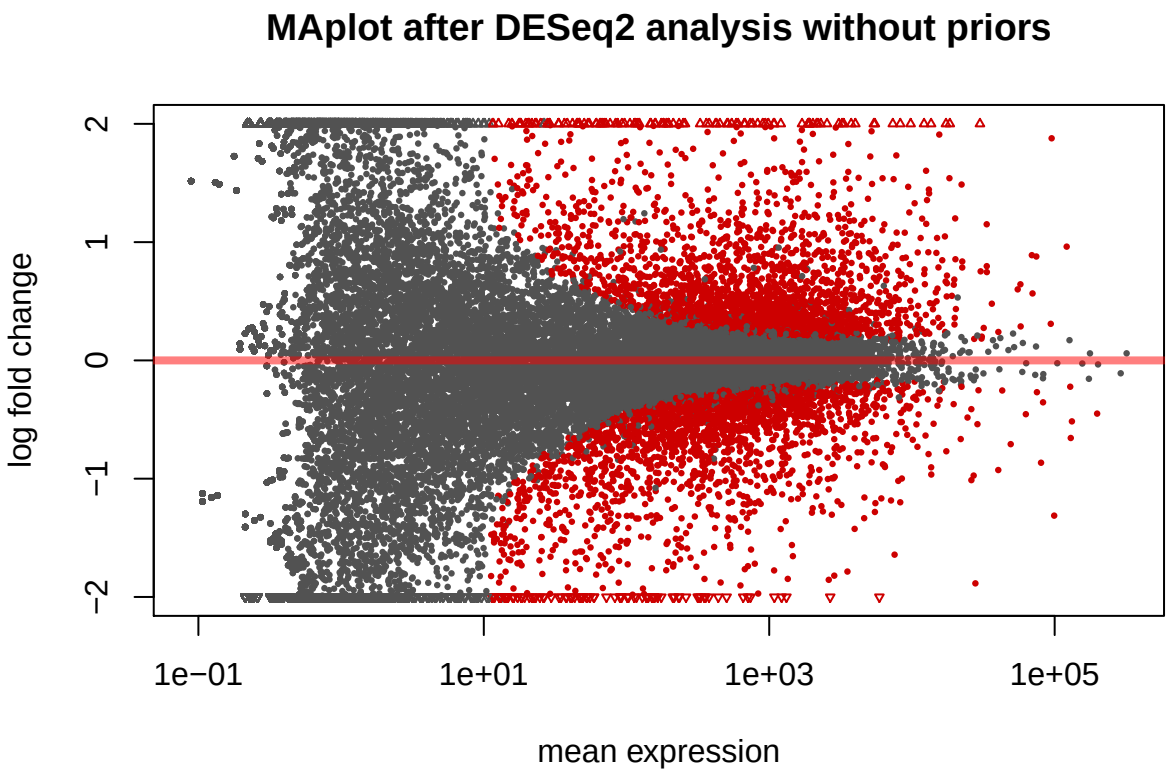
```
# color significant
points(log(resMLESig$baseMean,10), resMLESig$lfcMLE,
       col="red",
       pch=20,
       cex=0.25)

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




```
# SAME using built in capabilitiezs 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
```

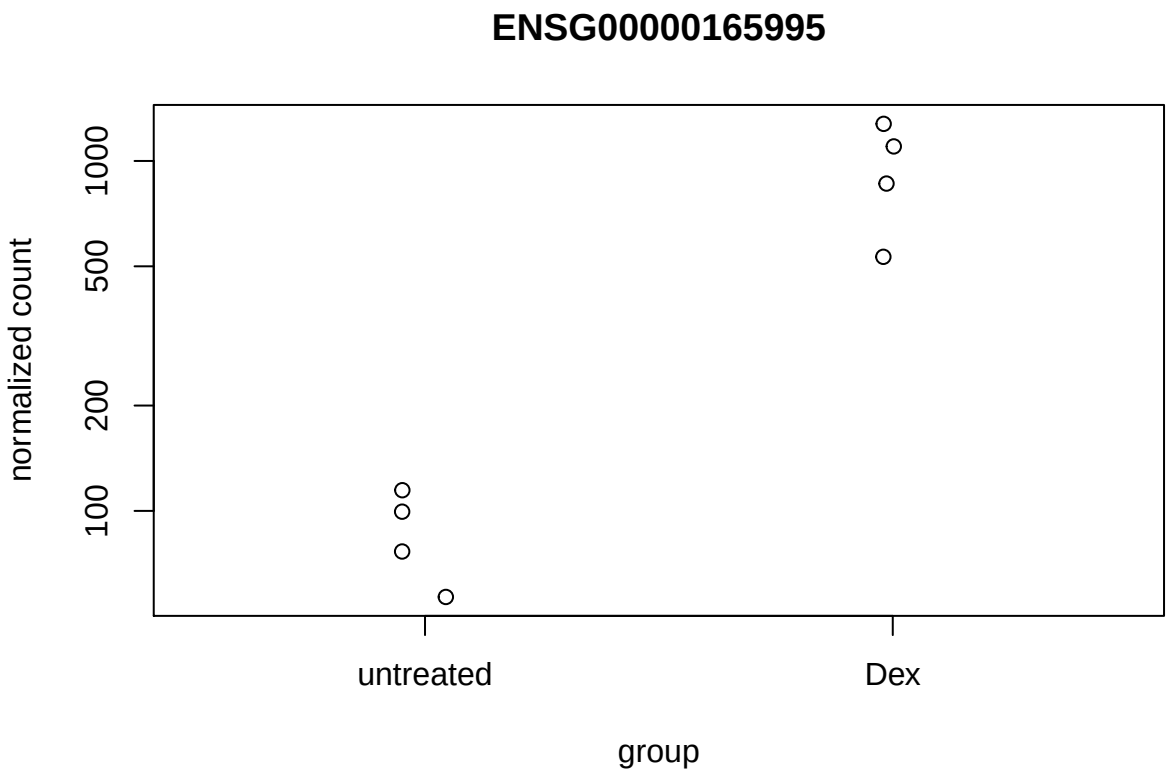
```
plotMA(resNoPrior, main="MAplot after DESeq2 analysis without priors",
       ylim=c(-2,2))
```



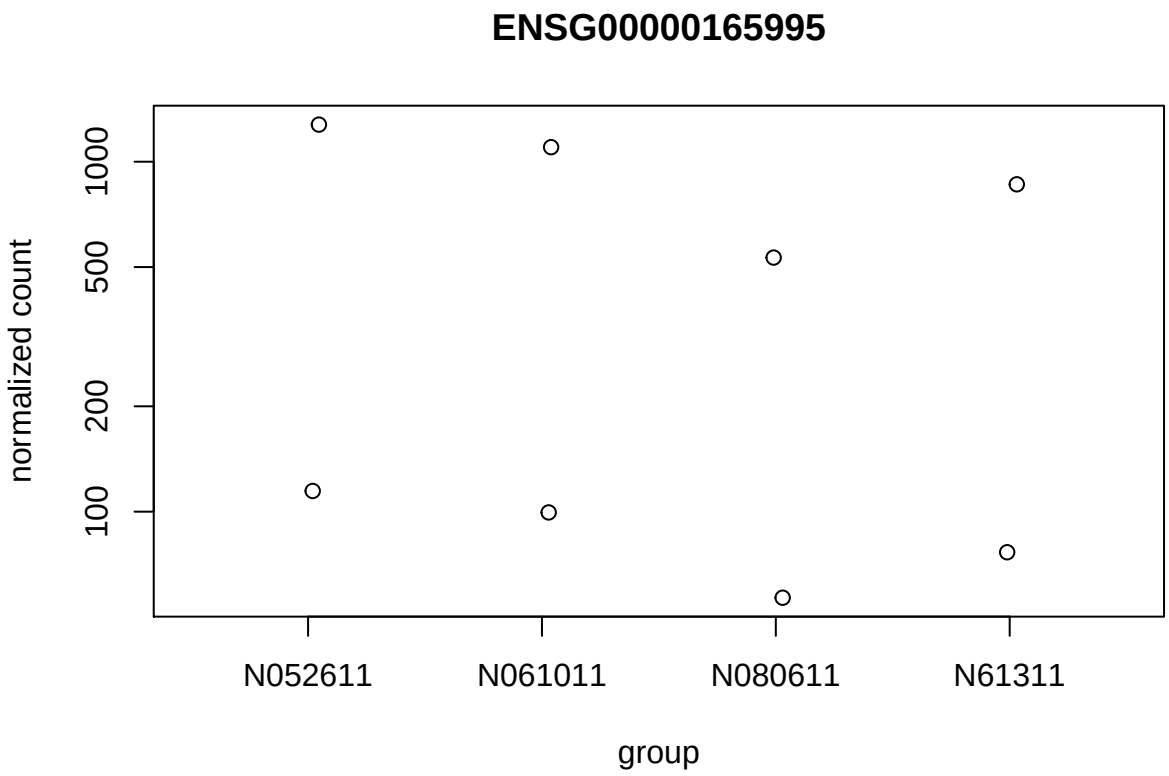
Plot counts

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

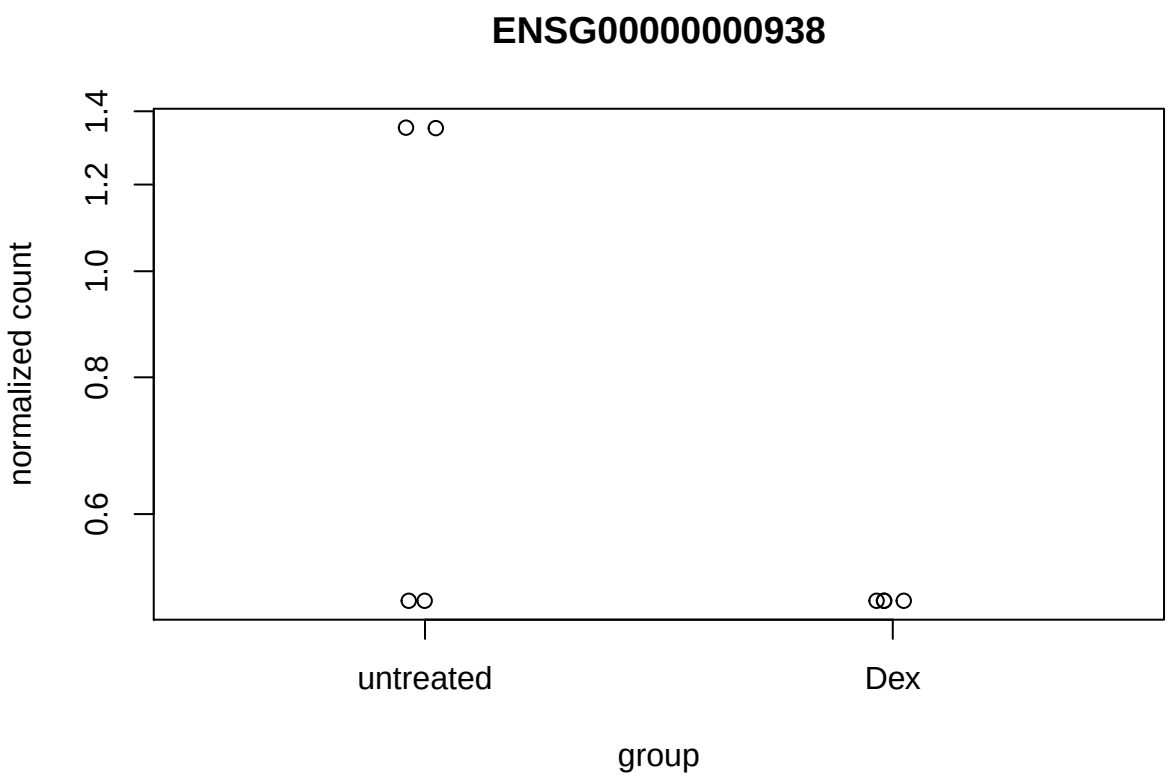
```
# simple plot for the best scoring gene (based on padj)
onegene <- rownames(res)[which.min(res$padj)]

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



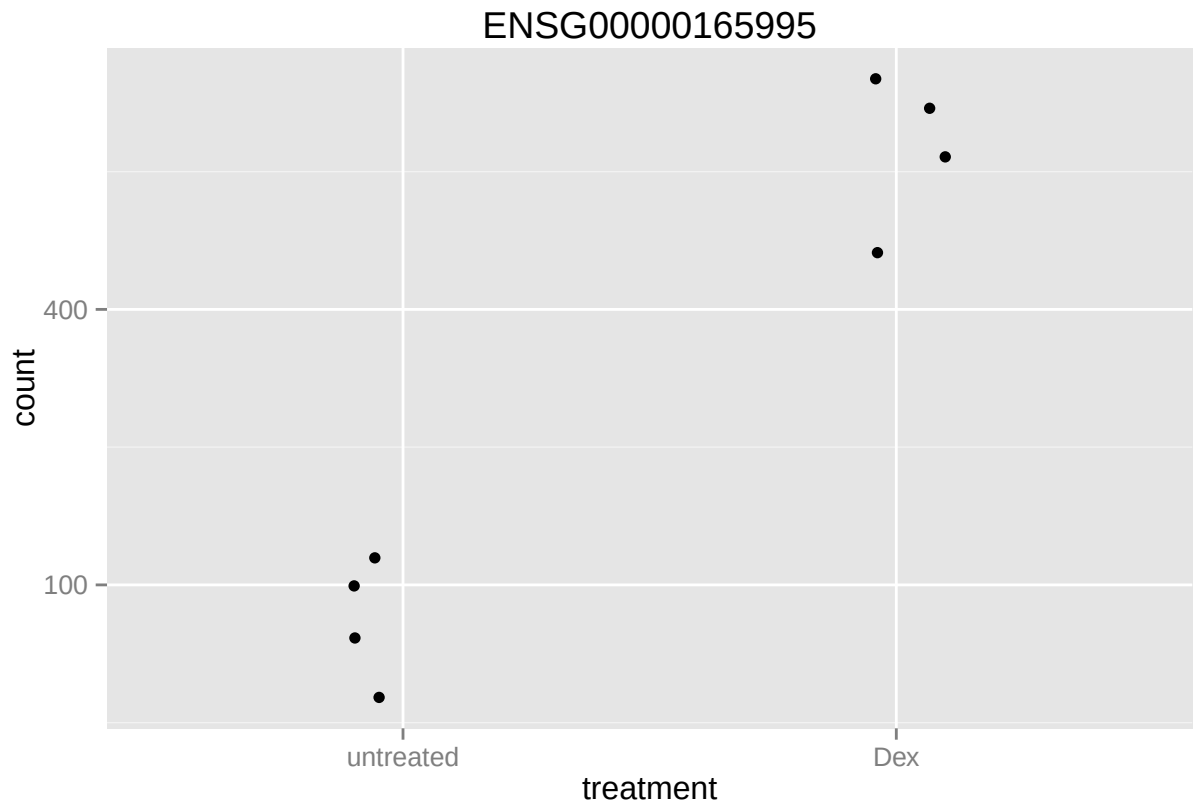

```
plotCounts(dds, gene=which.min(res$padj), intgroup="cells", main=onegene)
```



```
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), file="Dex_vs_untreated_results.csv")

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

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

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

```
##
## FALSE  TRUE
## 27918   812
```

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

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

```
table(resOrdered$padj<0.05)
```

```
##
## FALSE  TRUE
## 11180  3950
```

```
table( ( abs(resOrdered$log2FoldChange)>=1) & (resOrdered$padj<0.05) )
```

```
##
## FALSE  TRUE
## 27918   753
```

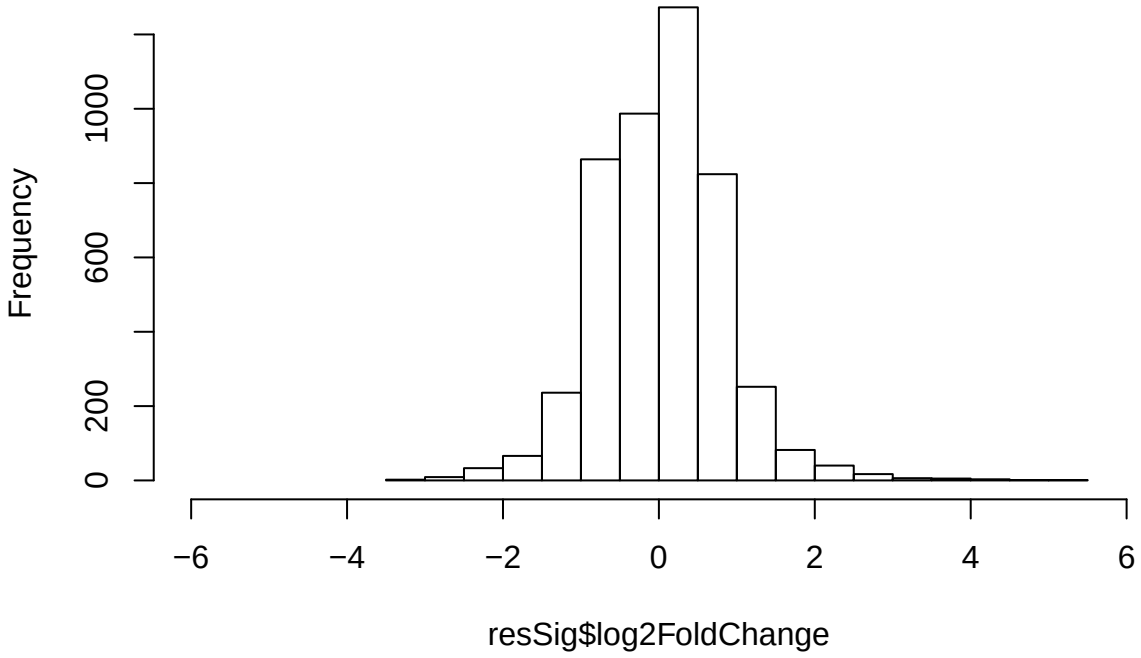
```
# extract significant subset for padj<0.1
resSig <- subset(resOrdered, padj < 0.1)

# preview results
resSig
```

```
## log2 fold change (MAP): treatment Dex vs untreated
## Wald test p-value: treatment Dex vs untreated
## DataFrame with 4701 rows and 6 columns
##                   baseMean log2FoldChange      lfcSE      stat
##                  <numeric>      <numeric>  <numeric> <numeric>
## ENSG00000165995   514.2841       3.321662  0.1307366  25.40728
## ENSG00000152583   985.5593       4.340812  0.1760858  24.65169
## ENSG00000120129  3325.4027       2.873150  0.1167734  24.60448
## ENSG00000101347 13616.9348       3.606558  0.1517550  23.76566
## ENSG00000189221  2294.7300       3.231983  0.1396172  23.14888
## ...                    ...            ...        ...       ...
## ENSG00000204055   16.16531     -0.7334479 0.33994399 -2.157555
## ENSG00000130312  418.38327      0.2439329 0.11306554  2.157447
## ENSG00000002933  528.89938      0.3013630 0.13972246  2.156868
## ENSG00000136709  906.46441      0.2265346 0.10504040  2.156643
## ENSG00000166479 2025.31592     -0.1926154 0.08932962 -2.156233
##                       pvalue          padj
##                    <numeric>     <numeric>
## ENSG00000165995 2.095444e-142 3.170407e-138
## ENSG00000152583 3.529479e-134 2.670051e-130
## ENSG00000120129 1.131211e-133 5.705072e-130
## ENSG00000101347 7.569341e-125 2.863103e-121
## ENSG00000189221 1.491976e-118 4.514718e-115
## ...                       ...           ...
## ENSG00000204055    0.03096243    0.09973633
## ENSG00000130312    0.03097088    0.09974232
## ENSG00000002933    0.03101593    0.09986614
## ENSG00000136709    0.03103352    0.09990153
## ENSG00000166479    0.03106550    0.09998320
```

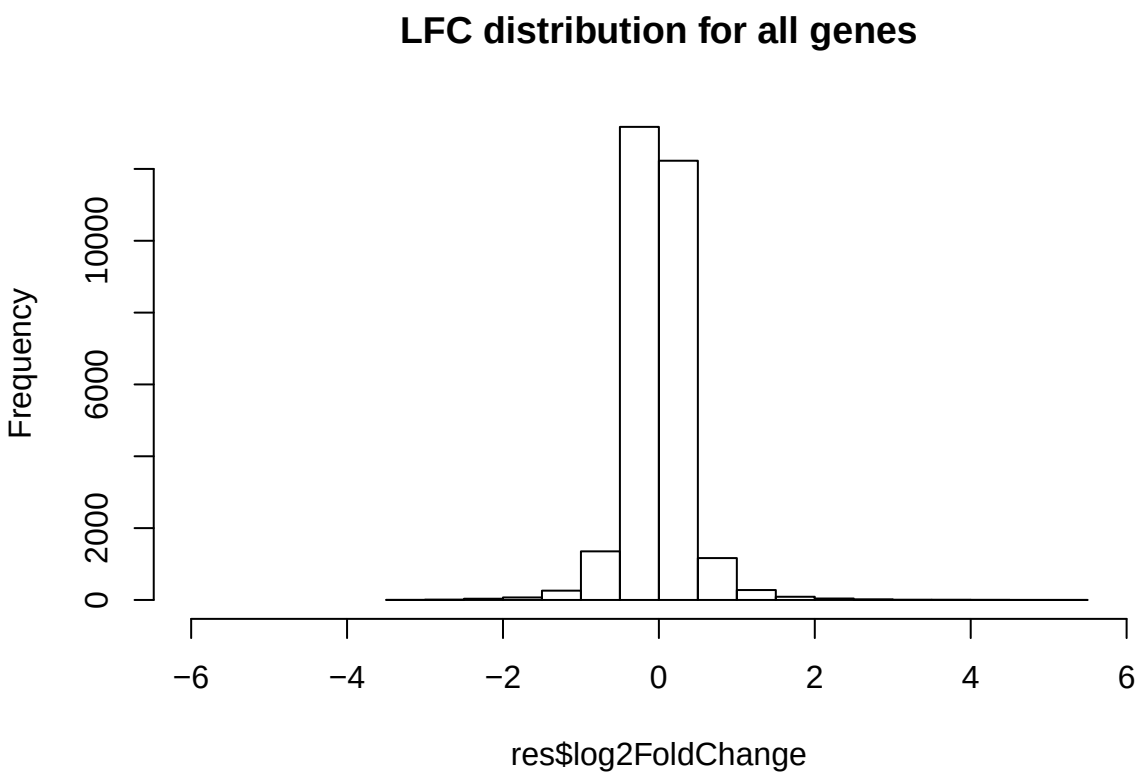
```r
# get dimentions
dim(resSig)
```

```
## [1] 4701    6
```

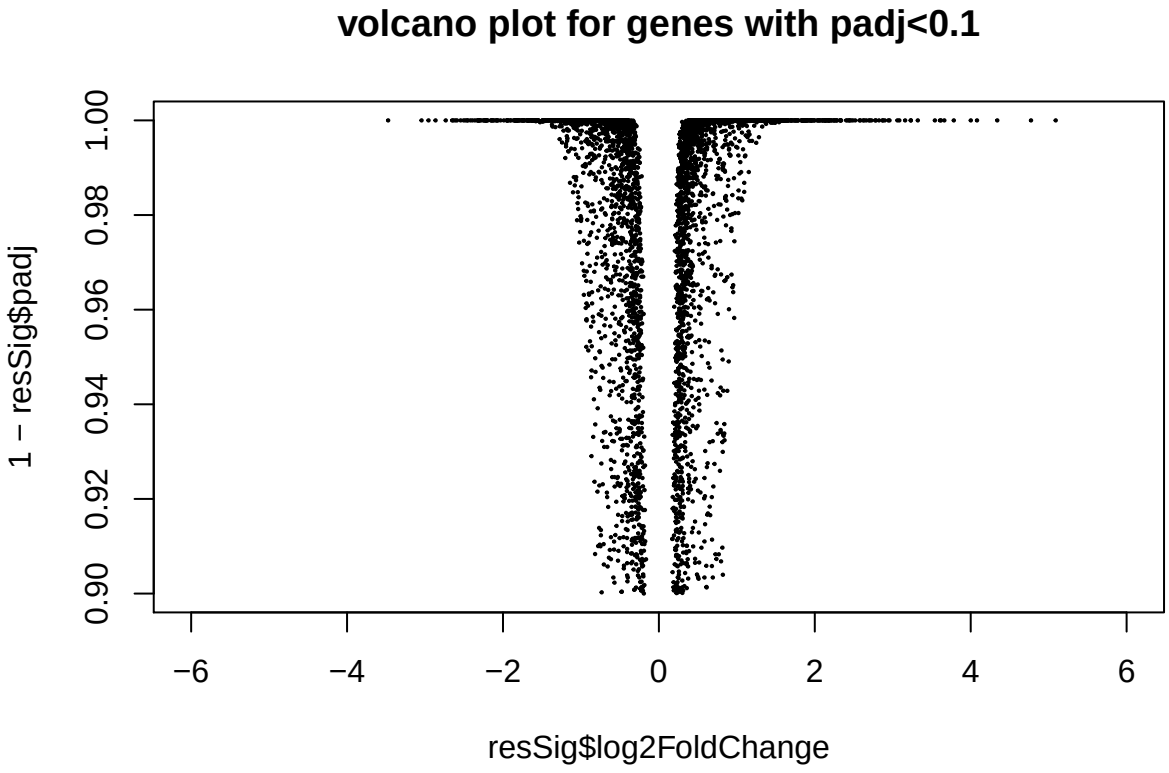
```r
# plot distribution of LFC in this subset
hist(resSig$log2FoldChange,
     xlim=c(-6,6),
     breaks=20,
     main="LFC distribution for genes with padj<0.1")
```



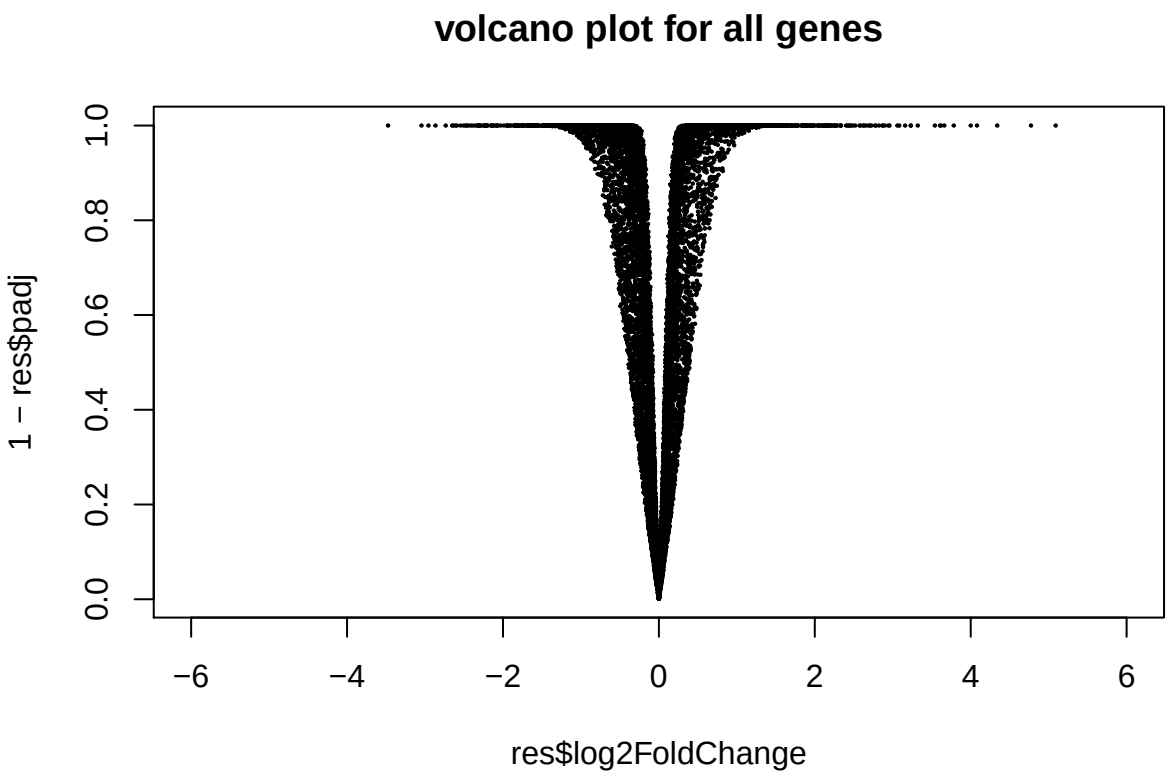

```r
hist(res$log2FoldChange,
     xlim=c(-6,6),
     breaks=20,
     main="LFC distribution for all genes")
```




```r
# volcano for the subset (using small dots)
plot(resSig$log2FoldChange, 1-resSig$padj,
     xlim=c(-6,6),
     main="volcano plot for genes with padj<0.1",
     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<0.05")
```

```
## [1] "# rows with abs(LFC) >=2 AND padj<0.05"
```

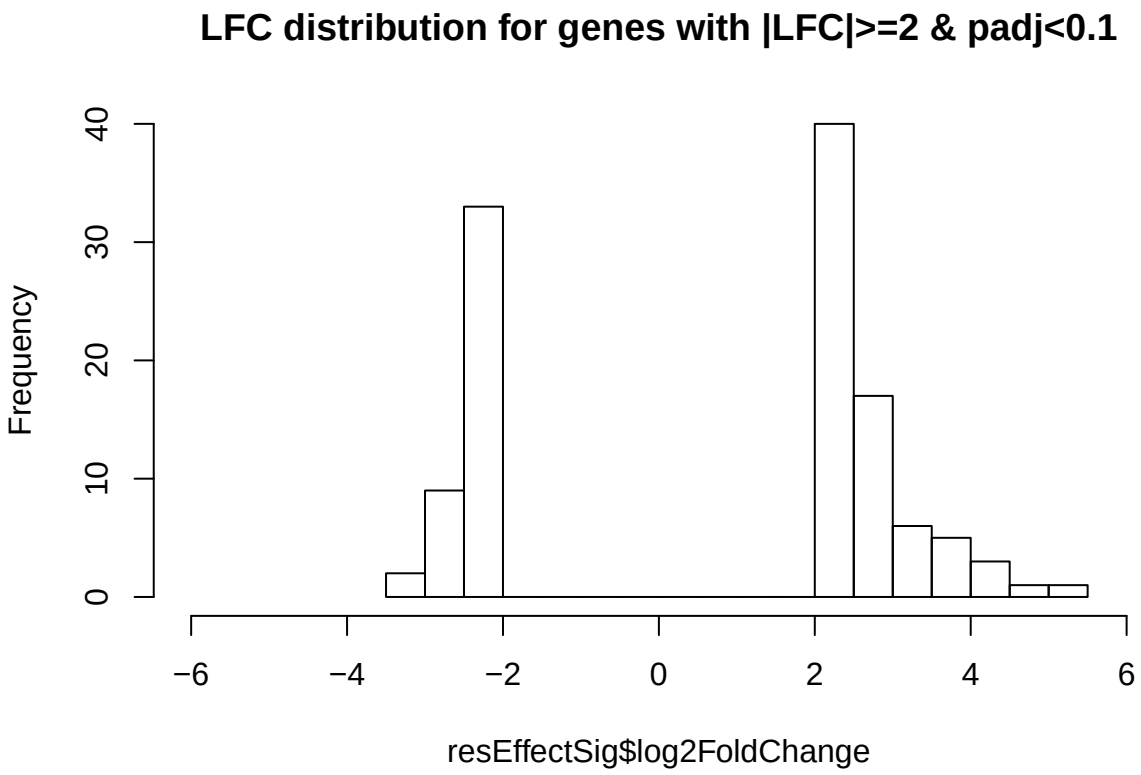
```
table( (abs(resOrdered$log2FoldChange)>=2 & resOrdered$padj<0.01) )
```

```
##
## FALSE  TRUE
## 28612   117
```

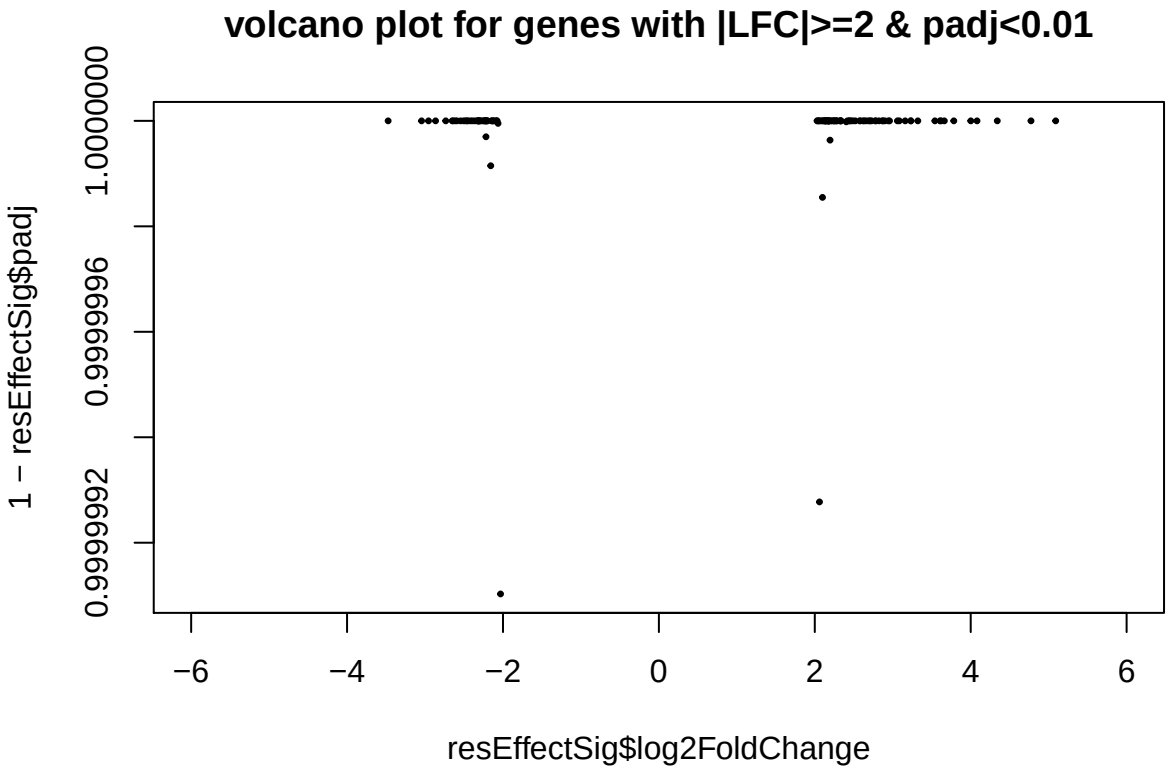
```
# make this a new table
resEffectSig <- subset(resOrdered, ( abs(resOrdered$log2FoldChange)>=2 & padj < 0.01 ))
dim(resEffectSig)
```

```
## [1] 117   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<0.1")
```




```r
# 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
## ENSG00000000003 662.0585590 447.3521198 888.3208158 389.460277
## ENSG00000000005          NA          NA          NA         NA
## ENSG00000000419 452.8914265 463.6440439 529.4967142 351.717591
## ENSG00000000457 254.5059353 227.1992356 288.0258265 166.832484
## ENSG00000000460  55.4085875  48.0864659  46.1068413  25.962791
## ENSG00000000938   0.2242071   0.1843136   0.2665927   0.142199
##                  SRR1039516  SRR1039517  SRR1039520  SRR1039521
## ENSG00000000003 1127.8985492 1032.2786636 729.6735829 577.8531689
## ENSG00000000005           NA          NA          NA          NA
## ENSG00000000419  523.2059349  725.4994127 385.5398906 462.5894101
## ENSG00000000457  292.2580087  353.3858775 235.6844418 246.5901055
## ENSG00000000460   55.3494530   65.0628273  56.0953888  57.0569724
## ENSG00000000938    0.2658388    0.2960058   0.2036141   0.1961786
```

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

```
##                  SRR1039508  SRR1039509 SRR1039512 SRR1039513   SRR1039516
## ENSG00000000003 0.001244826 0.016227787  0.0218152 0.01458686 0.0004660667
## ENSG00000000005          NA          NA         NA         NA           NA
## ENSG00000000419 0.218501698 0.240014949  0.2417025 0.18520521 0.0106669975
## ENSG00000000457 0.008011644 0.005404503  0.5170130 0.22277661 0.0552069344
## ENSG00000000460 0.001313744 0.068509195  1.6449549 0.14663913 0.2060011375
## ENSG00000000938 0.177461427 0.145887876  4.3045584 0.07178260 3.7595430482
##                  SRR1039517 SRR1039520  SRR1039521
## ENSG00000000003 0.006544443 0.01708741 0.014935045
## ENSG00000000005          NA         NA          NA
## ENSG00000000419 0.008249906 0.02995510 0.012188871
## ENSG00000000457 0.058638757 0.56126355 0.488143857
## ENSG00000000460 0.418521475 0.51179240 0.003959792
## ENSG00000000938 0.183993076 0.16116187 0.155279211
```

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: SummarizedExperiment
## dim: 57773 8
## exptData(0):
## assays(1): ''
## rownames(57773): ENSG00000000003 ENSG00000000005 ...
##   ENSG00000273492 ENSG00000273493
## rowData metadata column names(47): baseMean baseVar ... dispFit
##   rlogIntercept
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(3): cells treatment sizeFactor
```

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

```
## class: SummarizedExperiment
## dim: 57773 8
## exptData(0):
## assays(1): ''
## rownames(57773): ENSG00000000003 ENSG00000000005 ...
##   ENSG00000273492 ENSG00000273493
## rowData metadata column names(46): baseMean baseVar ...
##   dispGeneEst dispFit
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(3): cells treatment sizeFactor
```

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

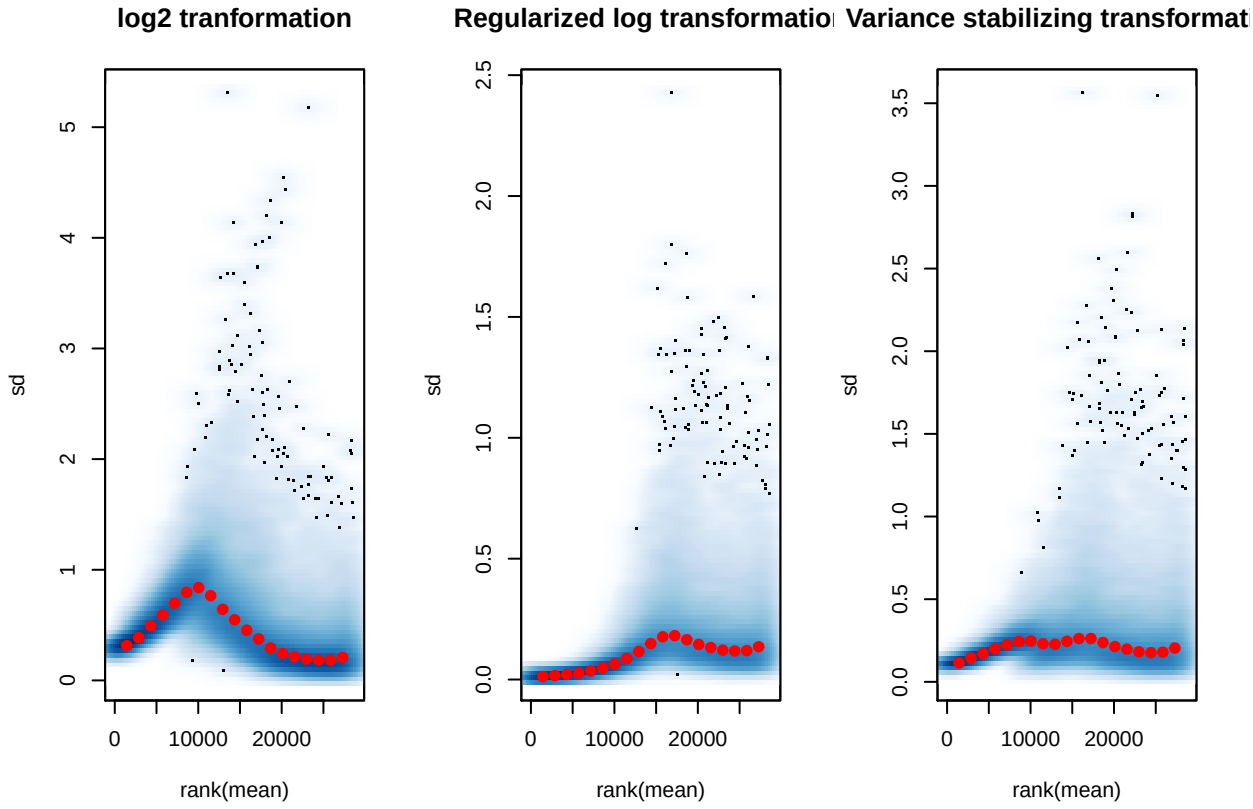
```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003   9.385815   9.111922   9.485295   9.293740   9.745669
## ENSG00000000005   0.000000   0.000000   0.000000   0.000000   0.000000
## ENSG00000000419   8.790046   9.016464   8.887312   8.928876   8.843568
## ENSG00000000457   7.979714   7.978258   7.915768   7.997564   7.953955
## ENSG00000000460   5.702651   5.735228   5.226608   5.562887   5.683232
## ENSG00000000938  -2.199605  -2.199305  -2.164748  -2.198557  -2.164706
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003   9.502458   9.584332   9.285416
## ENSG00000000005   0.000000   0.000000   0.000000
## ENSG00000000419   8.958822   8.761465   8.921048
## ENSG00000000457   7.986459   8.036356   7.958014
## ENSG00000000460   5.454195   5.909199   5.743980
## ENSG00000000938  -2.200320  -2.199371  -2.199459
```

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

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003   9.434437   9.016021   9.582868   9.294533   9.964458
## ENSG00000000005   4.063874   4.063874   4.063874   4.063874   4.063874
## ENSG00000000419   8.837919   9.175165   8.983583   9.045680   8.918437
## ENSG00000000457   8.156567   8.154439   8.062614   8.183129   8.118785
## ENSG00000000460   6.417750   6.465505   5.723628   6.213135   6.389237
## ENSG00000000938   4.063874   4.063874   4.388676   4.063874   4.388913
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003   9.608177   9.729533   9.282765
## ENSG00000000005   4.063874   4.063874   4.063874
## ENSG00000000419   9.089329   8.794405   9.033742
## ENSG00000000457   8.166373   8.239793   8.124620
## ENSG00000000460   6.066048   6.710763   6.477112
## ENSG00000000938   4.063874   4.063874   4.063874
```

Effects of transformations on the variance

```r
# library("vsn")
par(mfrow=c(1,3))
notAllZero <- (rowSums(counts(dds))>0)
meanSdPlot(log2(counts(dds,normalized=TRUE)[notAllZero,] + 1), main="log2 tranformation")
meanSdPlot(assay(rld[notAllZero,]), main="Regularized log transformation")
meanSdPlot(assay(vsd[notAllZero,]), main="Variance stabilizing transformation")
```

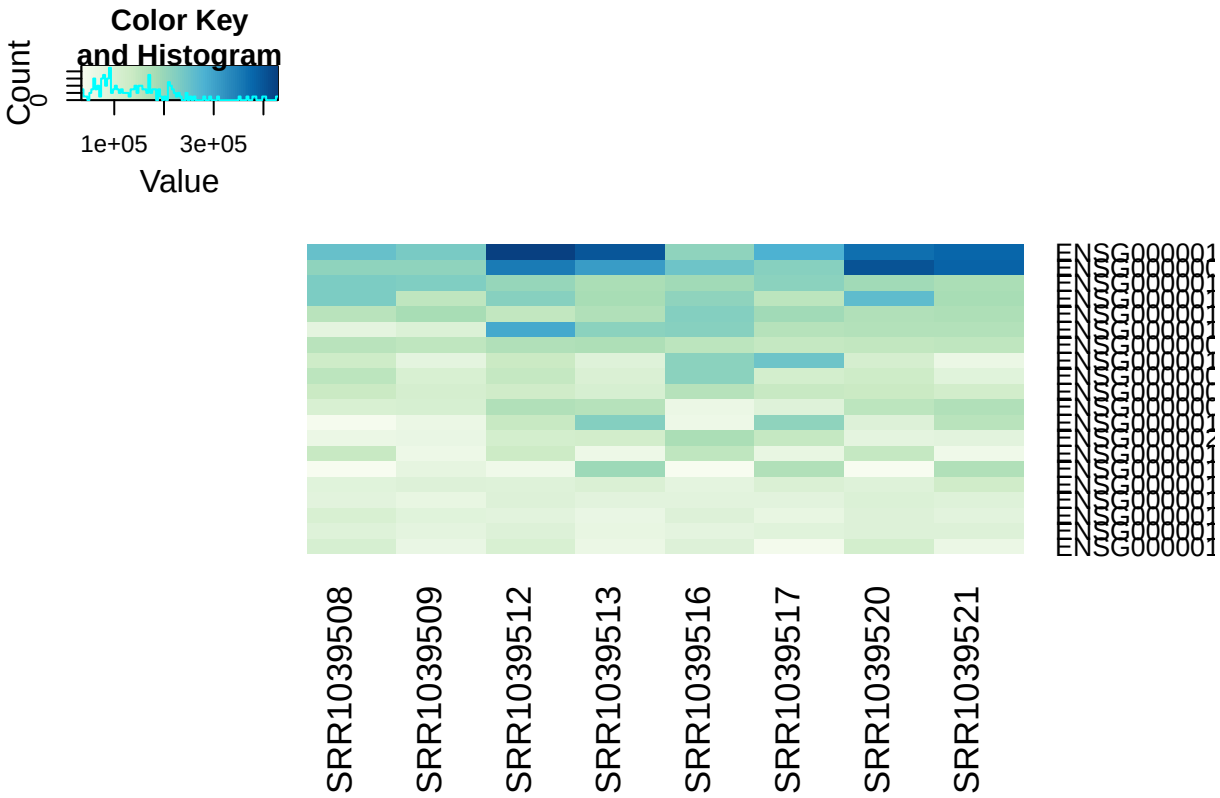



```
# legend: Standard deviation over mean. Per-gene standard deviation (taken across samples), against the
```

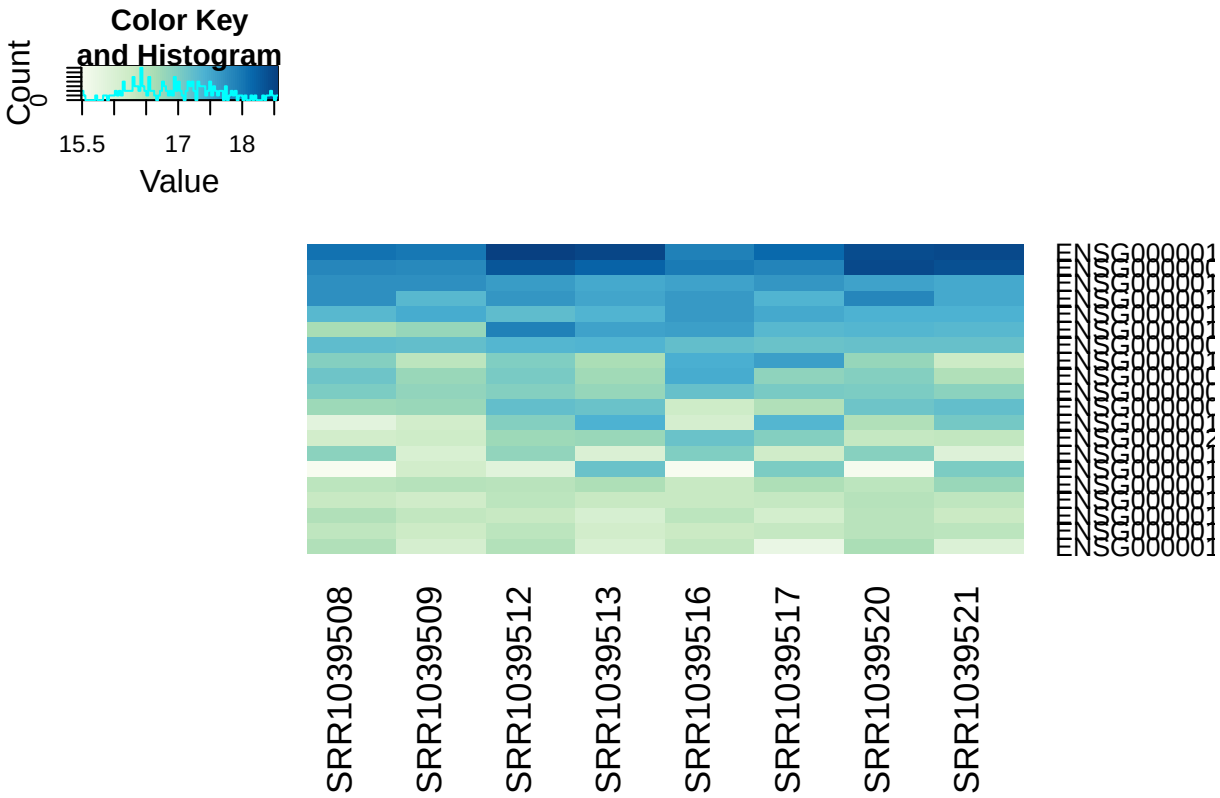
Data quality assessment by sample clustering and visualization

Heatmap of the count matrix

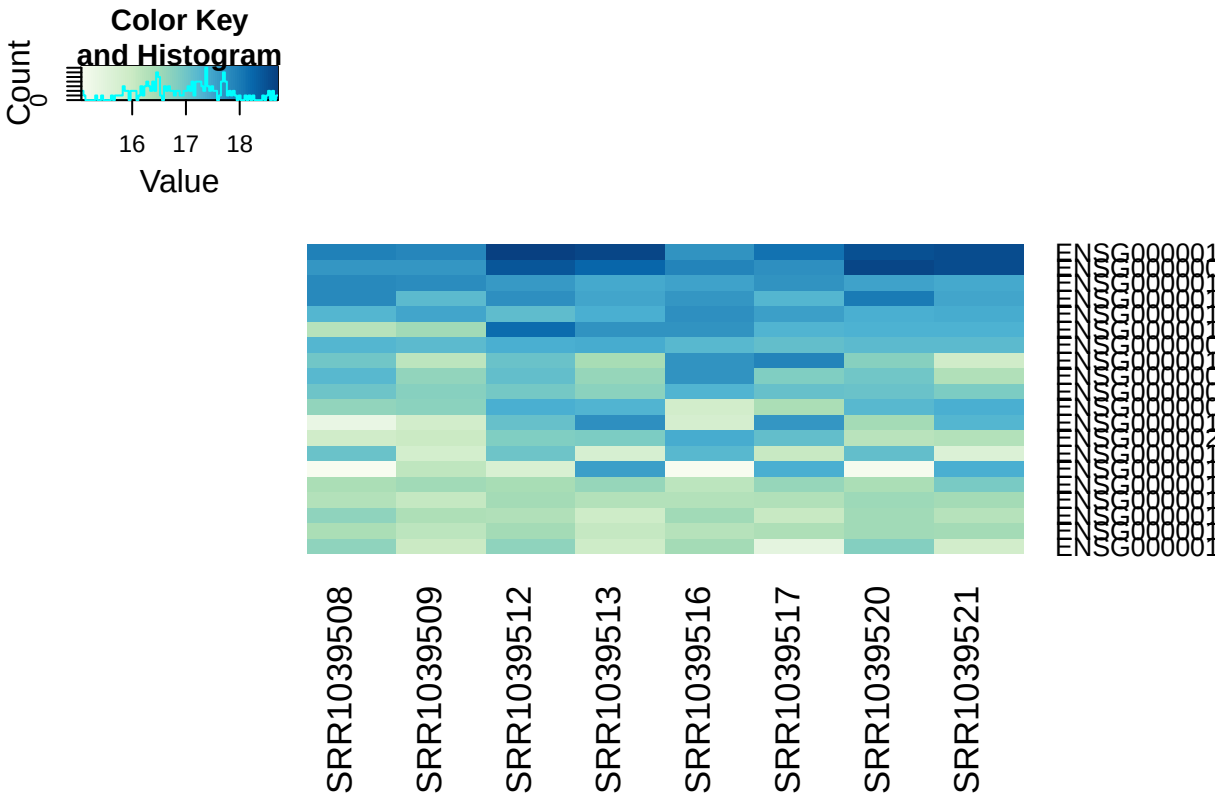
```
#library("RColorBrewer")
#library("gplots")

# 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)
          )
```



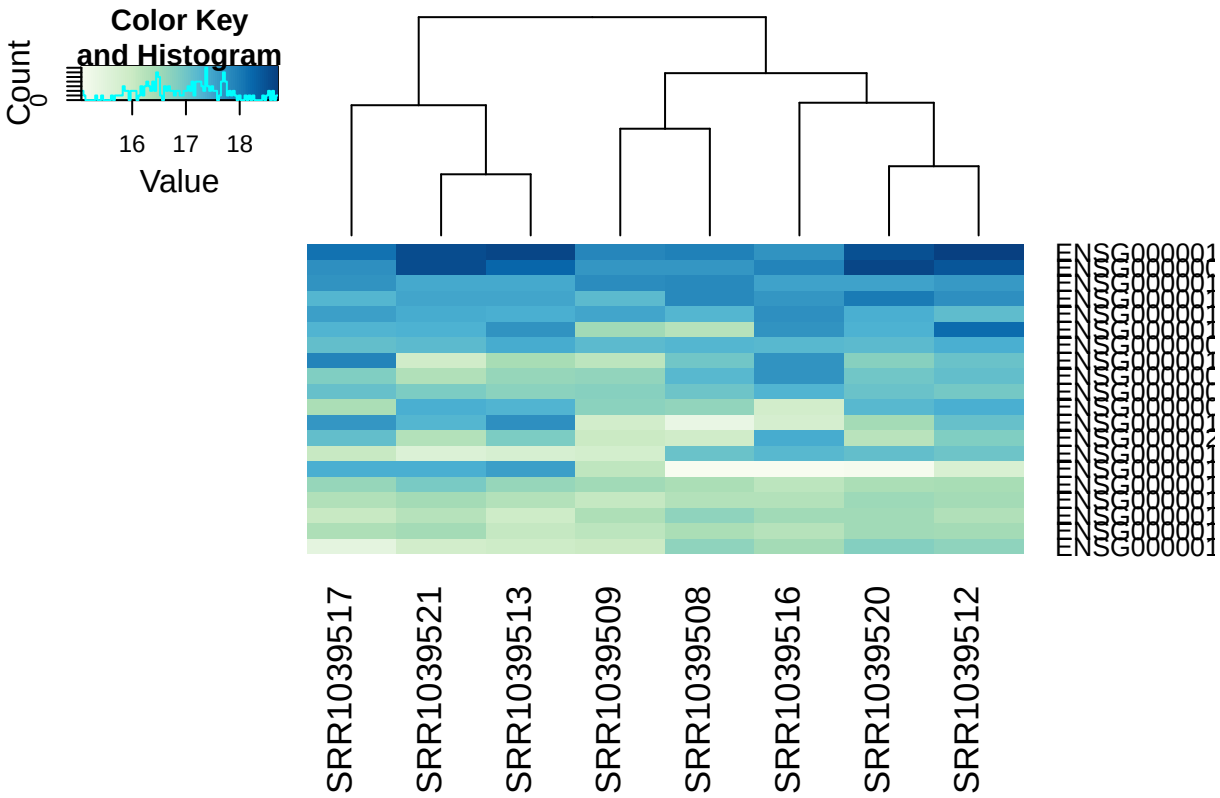

```
# show from Regularized log transformation
heatmap.2(assay(rld)[select,],
          col = hmcol,
          Rowv = FALSE,
          Colv = FALSE,
          scale="none",
          dendrogram="none",
          trace="none",
          margin=c(10, 6)
          )
```




```
# 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)
          )
```

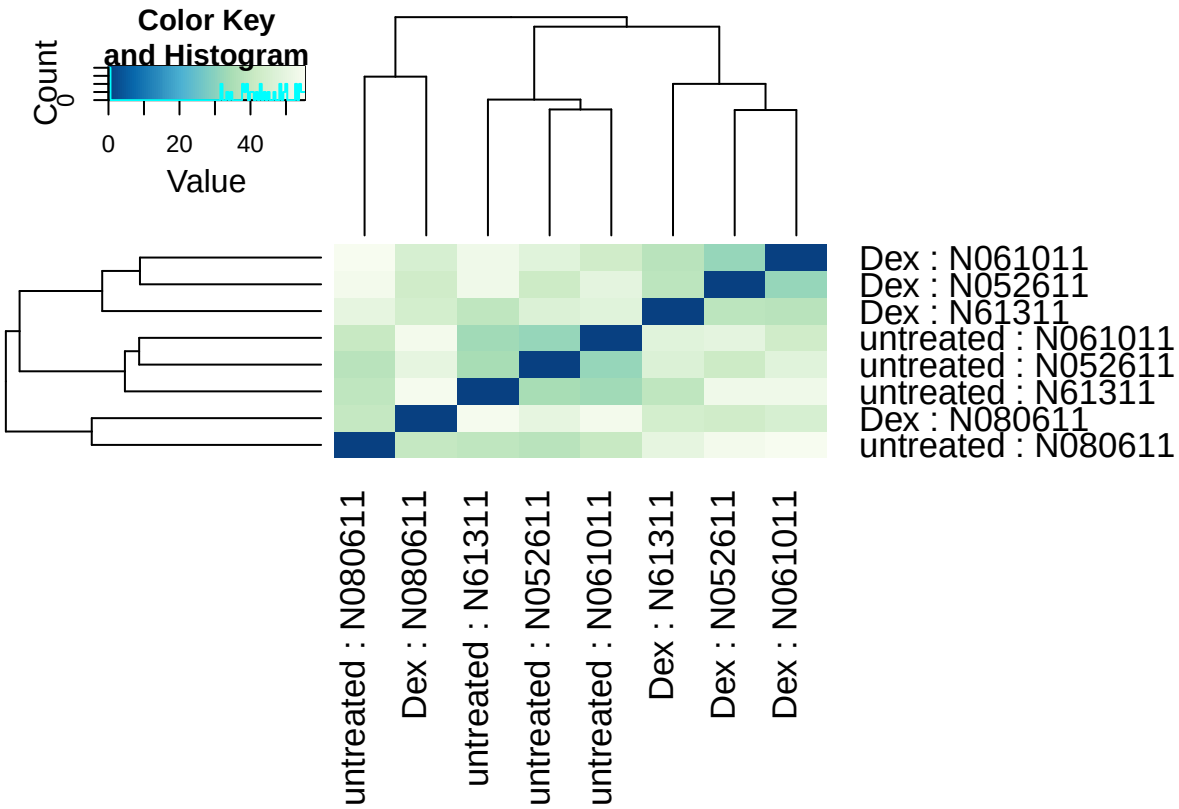



```
# show from Variance stabilizing transformation with sample clustering
heatmap.2(assay(vsd)[select,],
          col = hmcol,
          Rowv = FALSE,
          Colv = TRUE,
          scale="none",
          dendrogram="column",
          trace="none",
          margin=c(10, 6)
          )
```



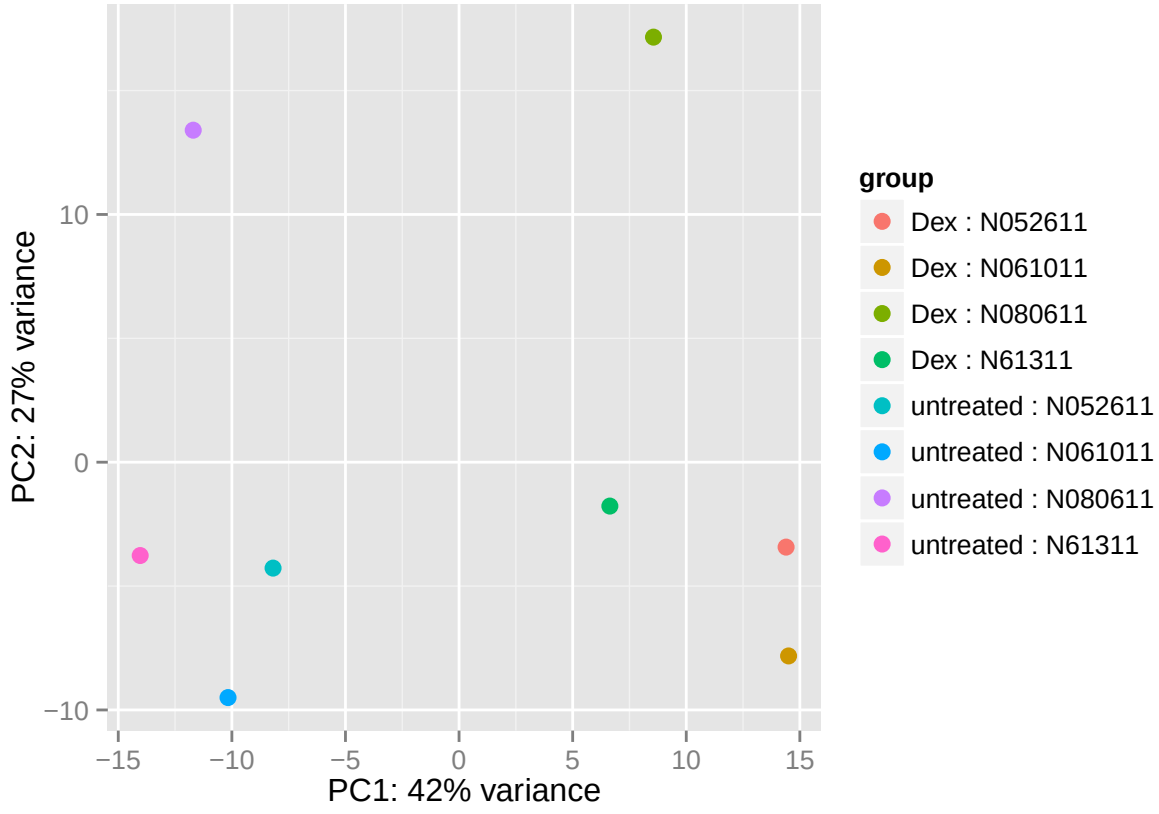
Heatmap of the sample-to-sample distances

```
# 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)
          )
```

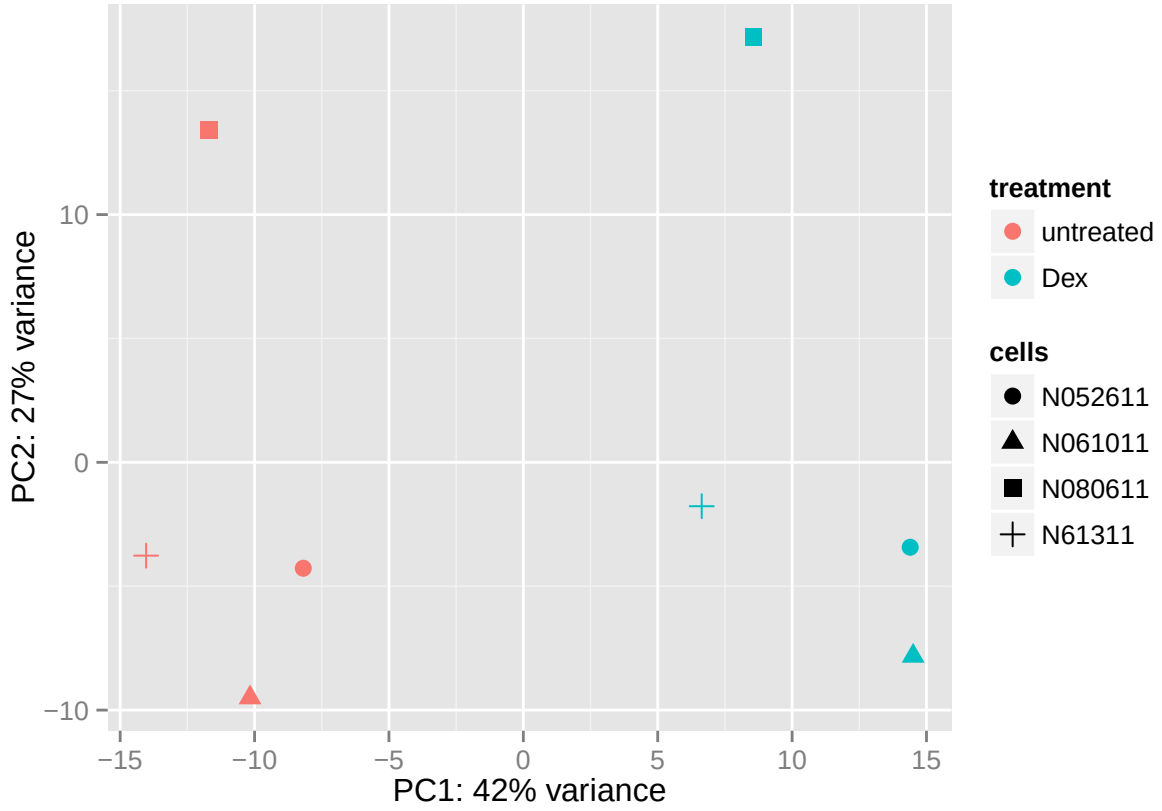


Principal component plot of the samples

```
# 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)

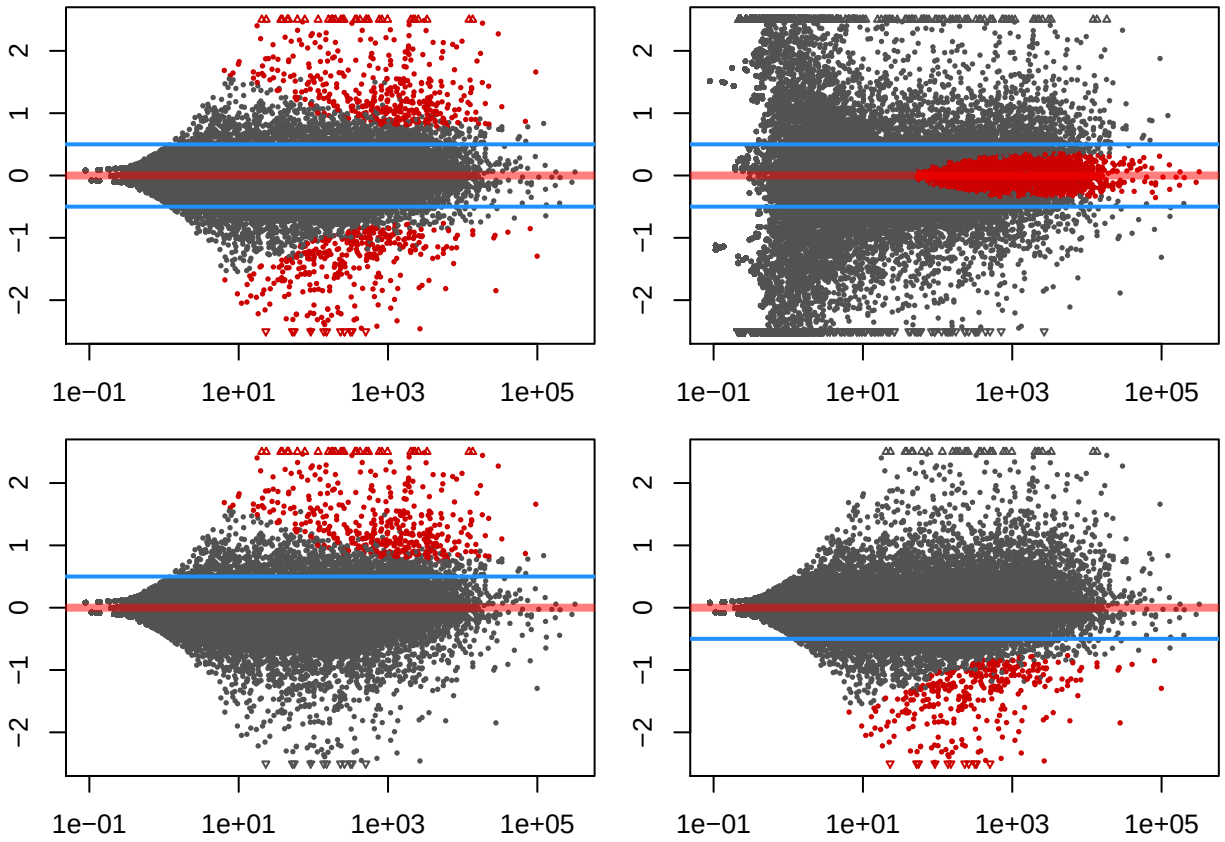
- greaterAbs - |beta| > lfcThreshold - tests are two-tailed
- lessAbs - |beta| < lfcThreshold - p values are the maximum of the upper and lower tests
- greater - beta > lfcThreshold
- less - beta < lfcThreshold

```
# 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 3.1.2 (2014-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=de_BE.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=de_BE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_BE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
```

```
## [8] methods   base
##
## other attached packages:
##  [1] gplots_2.16.0             RColorBrewer_1.1-2
##  [3] vsn_3.34.0                Biobase_2.26.0
##  [5] ggplot2_1.0.1             DESeq2_1.6.3
##  [7] RcppArmadillo_0.4.650.1.1 Rcpp_0.11.5
##  [9] GenomicRanges_1.18.4      GenomeInfoDb_1.2.5
## [11] IRanges_2.0.1             S4Vectors_0.4.0
## [13] BiocGenerics_0.12.1
##
## loaded via a namespace (and not attached):
##  [1] acepack_1.3-3.3       affy_1.44.0           affyio_1.34.0
##  [4] annotate_1.44.0       AnnotationDbi_1.28.2  base64enc_0.1-2
##  [7] BatchJobs_1.6         BBmisc_1.9            BiocInstaller_1.16.2
## [10] BiocParallel_1.0.3    bitops_1.0-6          brew_1.0-6
## [13] caTools_1.17.1        checkmate_1.5.2       cluster_2.0.1
## [16] codetools_0.2-11      colorspace_1.2-6      DBI_0.3.1
## [19] digest_0.6.8          evaluate_0.5.5        fail_1.2
## [22] foreach_1.4.2         foreign_0.8-63        formatR_1.1
## [25] Formula_1.2-1         gdata_2.13.3          genefilter_1.48.1
## [28] geneplotter_1.44.0    grid_3.1.2            gtable_0.1.2
## [31] gtools_3.4.1          Hmisc_3.15-0          htmltools_0.2.6
## [34] iterators_1.0.7       KernSmooth_2.23-14    knitr_1.9
## [37] labeling_0.3          lattice_0.20-31       latticeExtra_0.6-26
## [40] limma_3.22.7          locfit_1.5-9.1        MASS_7.3-40
## [43] munsell_0.4.2         nnet_7.3-9            plyr_1.8.1
## [46] preprocessCore_1.28.0 proto_0.3-10          reshape2_1.4.1
## [49] rmarkdown_0.5.1       rpart_4.1-9           RSQLite_1.0.0
## [52] scales_0.2.4          sendmailR_1.2-1       splines_3.1.2
## [55] stringr_0.6.2         survival_2.38-1       tools_3.1.2
## [58] XML_3.98-1.1          xtable_1.7-4          XVector_0.6.0
## [61] yaml_2.1.13           zlibbioc_1.12.0
```

 more at **http://www.bits.vib.be**