EdgeR_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 EdgeR vignette http://www.bioconductor.org/packages/release/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf where the HTSeq data is loaded in a EdgeR object for analysis.

Required packages

```
library("edgeR")
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)

# make sure the untreated samples are taken as control condition
sampleTable$treatment <- relevel(sampleTable$treatment, ref="untreated")

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
```

```
#   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 merged dataframe object. EdgeR does not provide a function to load separate HTSeq files, we need to merge them using flat R code. We merge only teh 'untreated' and 'Dex' files.

```
basedir <- "/media/bits/RNASeq_DATA"
setwd(basedir)

cntdir <- paste(basedir, "htseq_counts", sep="/")

# we take the file list from sampleTable$fileName
myfiles <- as.vector(sampleTable$fileName)
DT <- list()

# read each file as array element of DT and rename the last 2 cols
# we created a list of single sample tables
for (i in 1:length(myfiles) ) {
  infile = paste(cntdir, myfiles[i], sep = "/")
    DT[[myfiles[i]]] <- read.table(infile, header = F, stringsAsFactors = FALSE)
    cnts <- gsub("(.*)_all_counts.txt", "\\1", myfiles[i])
    colnames(DT[[myfiles[i]]]) <- c("ID", cnts)
}

# merge all elements based on first ID columns
data <- DT[[myfiles[1]]]

# inspect
head(data)
```

```
##                ID SRR1039508
## 1 ENSG00000000003        667
## 2 ENSG00000000005          0
## 3 ENSG00000000419        430
## 4 ENSG00000000457        256
## 5 ENSG00000000460         56
## 6 ENSG00000000938          0
```

```
# we now add each other table with the ID column as key
for (i in 2:length(myfiles)) {
    y <- DT[[myfiles[i]]]
    z <- merge(data, y, by = c("ID"))
    data <- z
}

# ID column becomes rownames
rownames(data) <- data$ID
# then disappears
data <- data[,-1]

# keep only rows with ENS IDs
data <- data[grep("^ENS", rownames(data), perl=TRUE, invert=FALSE), ]

# inspect and look at the top row names!
head(data)
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003        667        434        862        401       1133
## ENSG00000000005          0          0          0          0          0
## ENSG00000000419        430        488        556        334        529
## ENSG00000000457        256        226        276        174        288
## ENSG00000000460         56         52         29         30         63
## ENSG00000000938          0          0          1          0          1
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003       1050        750        562
## ENSG00000000005          0          0          0
## ENSG00000000419        719        378        468
## ENSG00000000457        358        248        235
## ENSG00000000460         53         68         56
## ENSG00000000938          0          0          0
```

```
#                SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
# ENSG00000000003        667        434        862        401       1133       1050        750
# ENSG00000000005          0          0          0          0          0          0          0
# ENSG00000000419        430        488        556        334        529        719        378
# ENSG00000000457        256        226        276        174        288        358        248
# ENSG00000000460         56         52         29         30         63         53         68
# ENSG00000000938          0          0          1          0          1          0          0
#                SRR1039521
# ENSG00000000003        562
# ENSG00000000005          0
# ENSG00000000419        468
# ENSG00000000457        235
# ENSG00000000460         56
# ENSG00000000938          0

tail(data)
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000273486         13         11         23          6         20
## ENSG00000273487          3          4          0          9          6
## ENSG00000273488          5          3          5          3          3
## ENSG00000273489          0          0          0          2          1
## ENSG00000273492          0          0          1          0          0
## ENSG00000273493          0          0          0          0          0
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000273486         29         12         11
## ENSG00000273487          4          3          8
## ENSG00000273488         10          6         11
## ENSG00000273489          0          0          0
## ENSG00000273492          0          0          0
## ENSG00000273493          0          0          0
```

```
#                SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
# ENSG00000273486         13         11         23          6         20         29         12
# ENSG00000273487          3          4          0          9          6          4          3
# ENSG00000273488          5          3          5          3          3         10          6
# ENSG00000273489          0          0          0          2          1          0          0
# ENSG00000273492          0          0          1          0          0          0          0
# ENSG00000273493          0          0          0          0          0          0          0
#                SRR1039521
# ENSG00000273486         11
# ENSG00000273487          8
# ENSG00000273488         11
# ENSG00000273489          0
# ENSG00000273492          0
# ENSG00000273493          0
# destroy useless objects
rm(y,z,DT)
```

The GLM approach with paired Model and Blocking

The difference associated with cell line are not our primary interrest and can be removed from the equation by using a paired design and GLM models (blocking).

```r
# we create a new DGE object with counts
y <- DGEList(counts=data, group=sampleTable$treatment)

colnames(y)
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```r
# define design factors
cells <- factor(sampleTable$cells)
treatment <- factor(sampleTable$treatment)
design <- model.matrix(~cells+treatment)
```

Filter out low expression genes

These genes introduce noise in teh model and are not interesting due to their extreme variance. Better get rid of them now.

```r
# how many genes & samples in the full data
dim(y)
```

```
## [1] 57773     8
```

```r
# [1] 57773     8

# Filter out genes with too few counts
# REM: 100cpm for the full row sum ~5 reads
keep <- rowSums(cpm(y)>100) >=2
y2 <- y[keep,,keep.lib.sizes=FALSE]

# the last parameter 'keep.lib.sizes=FALSE' was added
# to ignore removed genes in the new library size computation

# how many genes & samples remain after filtering
dim(y2)
```

```
## [1] 2176    8
```

```r
# [1] 2176    8

# estimate Common dispersion and show result
y <- estimateGLMCommonDisp(y, design)
y$common.dispersion
```

```
## [1] 0.01189821
```

```r
# before filtering: [1] 0.01189838

y2 <- estimateGLMCommonDisp(y2, design)
y2$common.dispersion
```

```
## [1] 0.01186341
```

```
# [1] 0.01186341

# we need either Common or Trended dispersion to compute the Tagwise dispersion
# alt: estimate Trended dispersion and show result
y <- estimateGLMTrendedDisp(y, design)
```

```
## Loading required package: splines
```

```
head(y$trended.dispersion)
```

```
## [1] 0.007756202 0.715681255 0.006269946 0.007801994 0.046244987 0.586319070
```

```
# before filtering: [1] 0.007756180 0.715691079 0.006269901 0.007801945 0.046245273 0.586326434

y2 <- estimateGLMTrendedDisp(y2, design)
head(y2$trended.dispersion)
```

```
## [1] 0.012350948 0.013354192 0.008984367 0.014751429 0.009823789 0.008809909
```

```
# [1] 0.012350948 0.013354192 0.008984367 0.014751429 0.009823789 0.008809909

# estimate Tagwise dispersion and show result
y <- estimateGLMTagwiseDisp(y, design)
head(y$tagwise.dispersion)
```

```
## [1] 0.007073360 0.715681255 0.007713469 0.008693132 0.034085004 0.586319070
```

```
# before filtering: [1] 0.007073358 0.715691079 0.007713463 0.008693123 0.034085032 0.586326434

y2 <- estimateGLMTagwiseDisp(y2, design)
head(y2$tagwise.dispersion)
```

```
## [1] 0.01026683 0.00974898 0.00971448 0.01034537 0.00847316 0.01115426
```

```
# [1] 0.010266828 0.009748980 0.009714480 0.010345372 0.008473159 0.011154258

# normalize samples
y <- calcNormFactors(y)
y2 <- calcNormFactors(y2)

# review results
y$samples
```

```
##                 group lib.size norm.factors
## SRR1039508 untreated 18104038    1.0707347
## SRR1039509       Dex 16545934    1.0411533
## SRR1039512 untreated 22630840    0.9812414
## SRR1039513       Dex 13714148    0.9441635
## SRR1039516 untreated 21747061    1.0258996
## SRR1039517       Dex 27823823    0.9646121
## SRR1039520 untreated 17140081    1.0266804
## SRR1039521       Dex 19170765    0.9529882
```

```
# before filtering
#                 group lib.size norm.factors
# SRR1039508 untreated 18104038    1.0707347
# SRR1039509       Dex 16545934    1.0411533
# SRR1039512 untreated 22630840    0.9812414
# SRR1039513       Dex 13714148    0.9441635
# SRR1039516 untreated 21747061    1.0258996
# SRR1039517       Dex 27823823    0.9646121
# SRR1039520 untreated 17140081    1.0266804
# SRR1039521       Dex 19170765    0.9529882

y2$samples
```

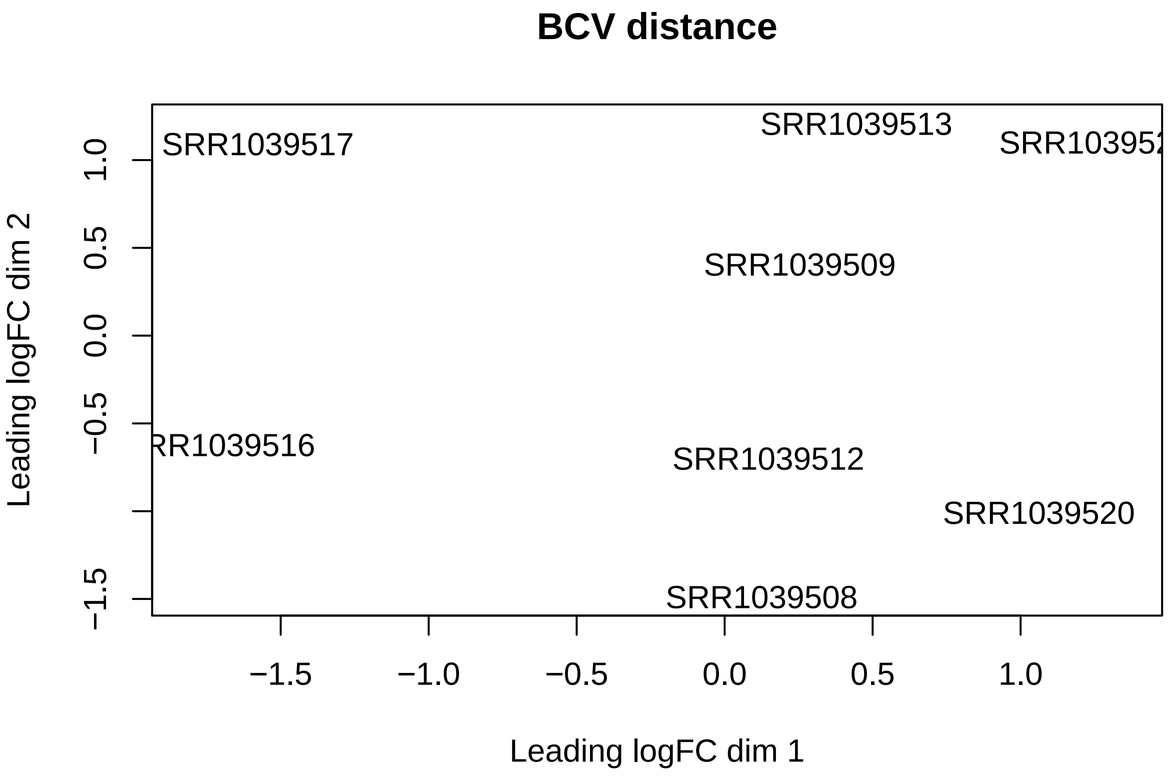
```
##                group lib.size norm.factors
## SRR1039508 untreated 12962520    1.0487639
## SRR1039509       Dex 11937040    1.0563501
## SRR1039512 untreated 16739654    0.9615356
## SRR1039513       Dex 10232283    0.9585992
## SRR1039516 untreated 15769663    1.0190399
## SRR1039517       Dex 20543889    0.9899991
## SRR1039520 untreated 12505686    0.9930354
## SRR1039521       Dex 14199563    0.9775101
```

```
# after filtering:
#                group lib.size norm.factors
# SRR1039508 untreated 12962520    1.0487639
# SRR1039509       Dex 11937040    1.0563501
# SRR1039512 untreated 16739654    0.9615356
# SRR1039513       Dex 10232283    0.9585992
# SRR1039516 untreated 15769663    1.0190399
# SRR1039517       Dex 20543889    0.9899991
# SRR1039520 untreated 12505686    0.9930354
# SRR1039521       Dex 14199563    0.9775101
```

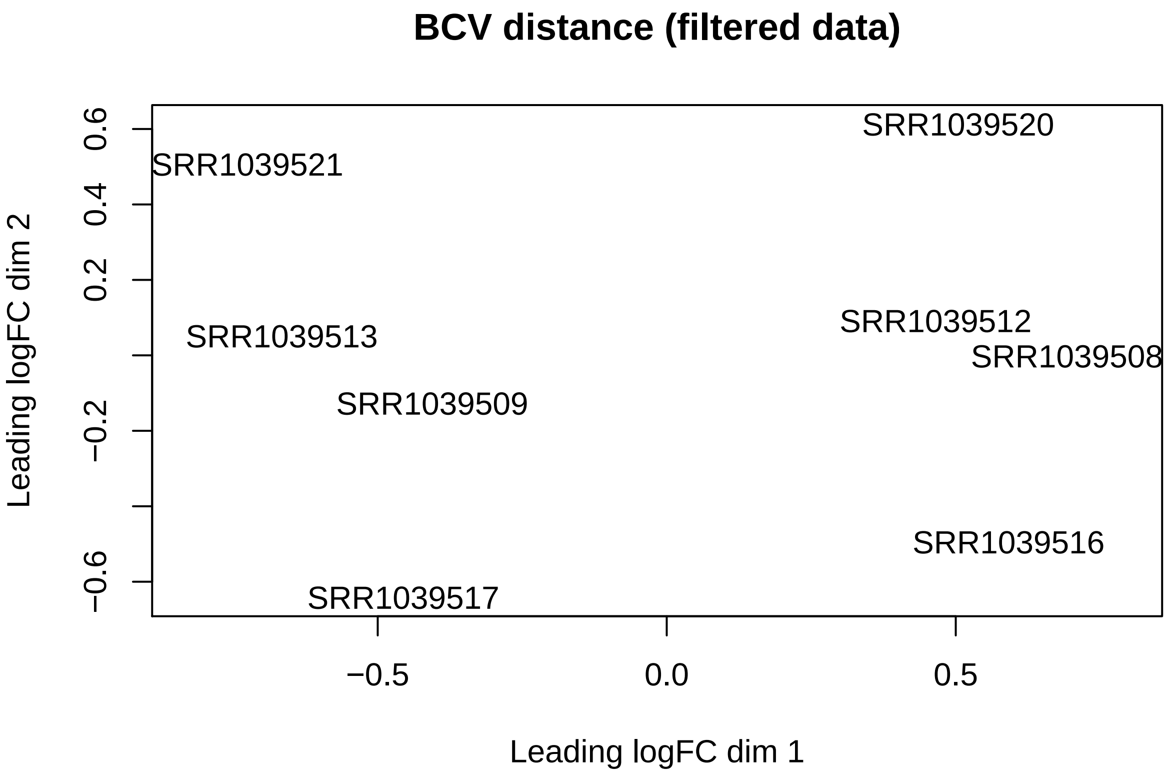
Estimate dispersion

When a negative binomial model is fitted, one needs to estimate the BCV(s) which is equivalent to estimating the dispersion of the data.

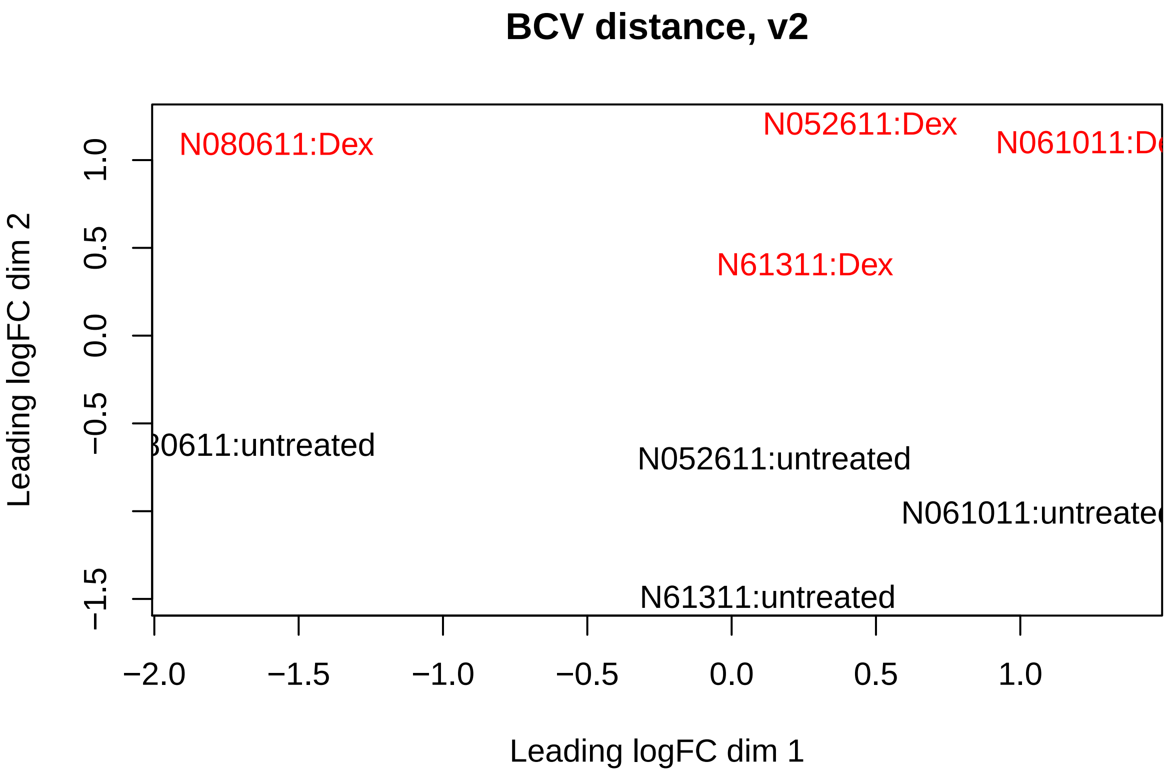
```
# estimate variance
# MDS plots that shows distances,
#     in terms of biological coefficient of variation (BCV), between samples
plotMDS(y, main="BCV distance")
```



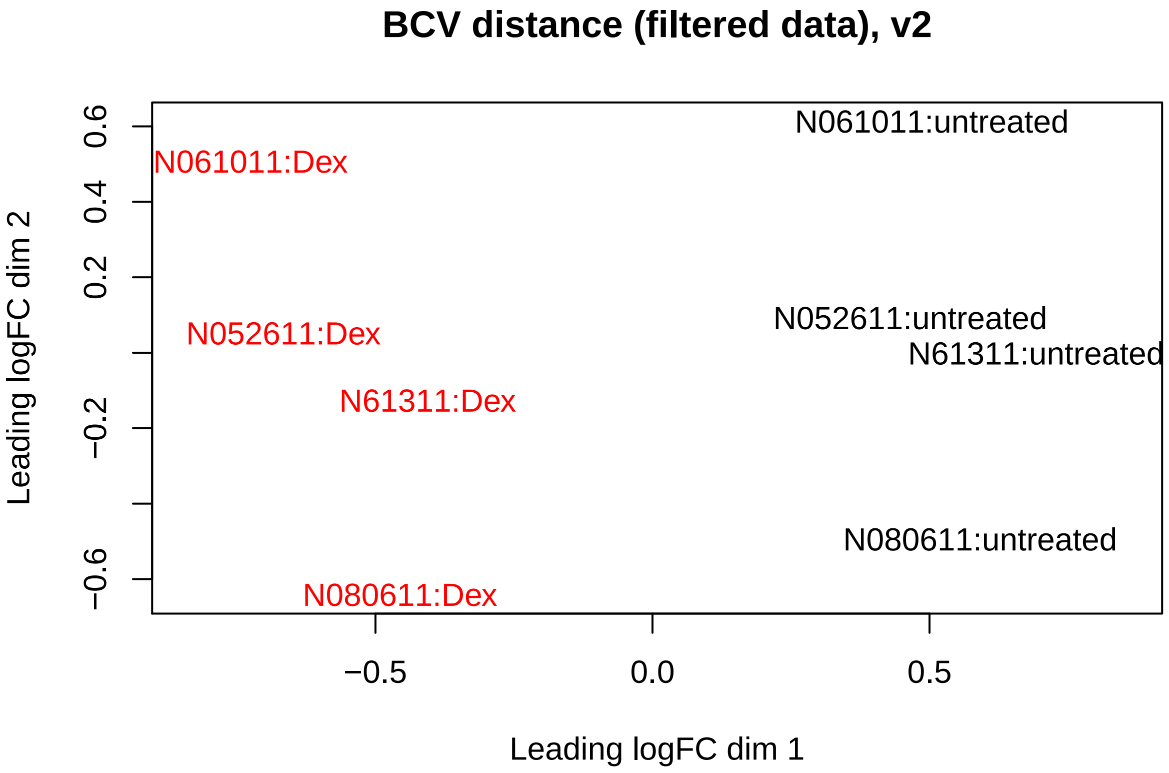
```
plotMDS(y2, main="BCV distance (filtered data)")
```



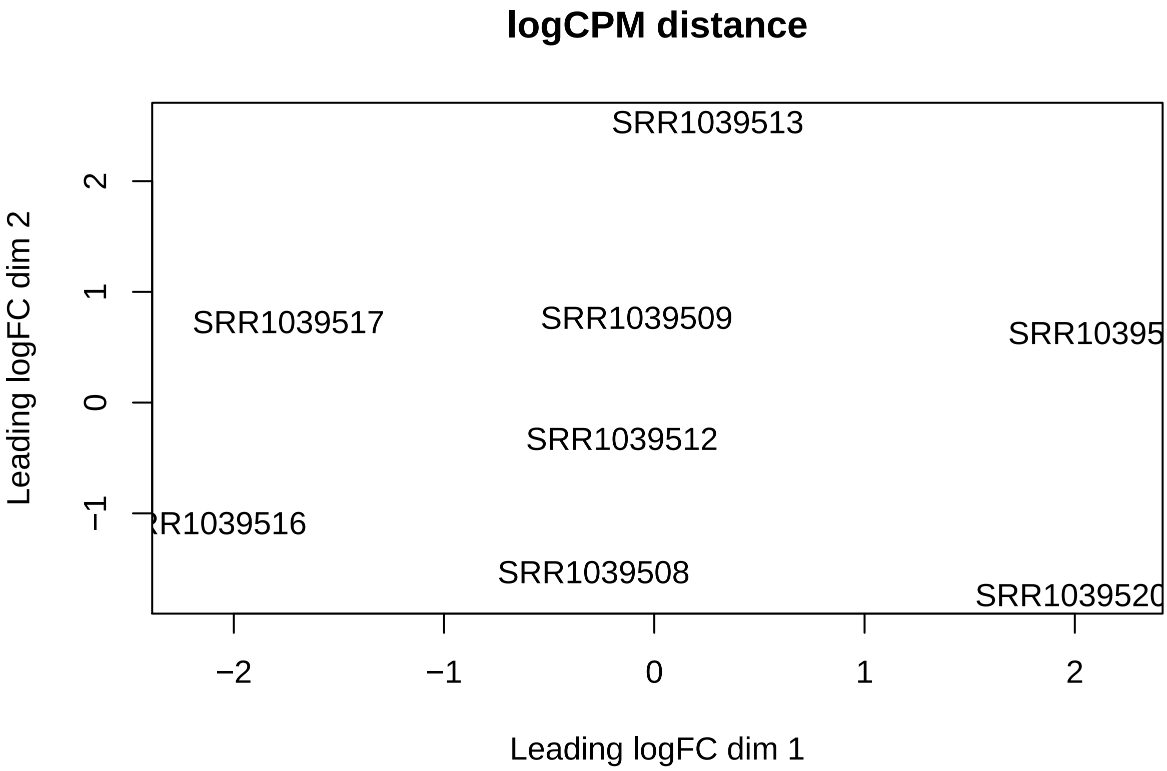
```
# a bit nicer plot
plotMDS(y, main="BCV distance, v2",
        col=c(rep(c("black","red"),4)),
        labels=paste(sampleTable$cells, sampleTable$treatment, sep=":")
          )
```



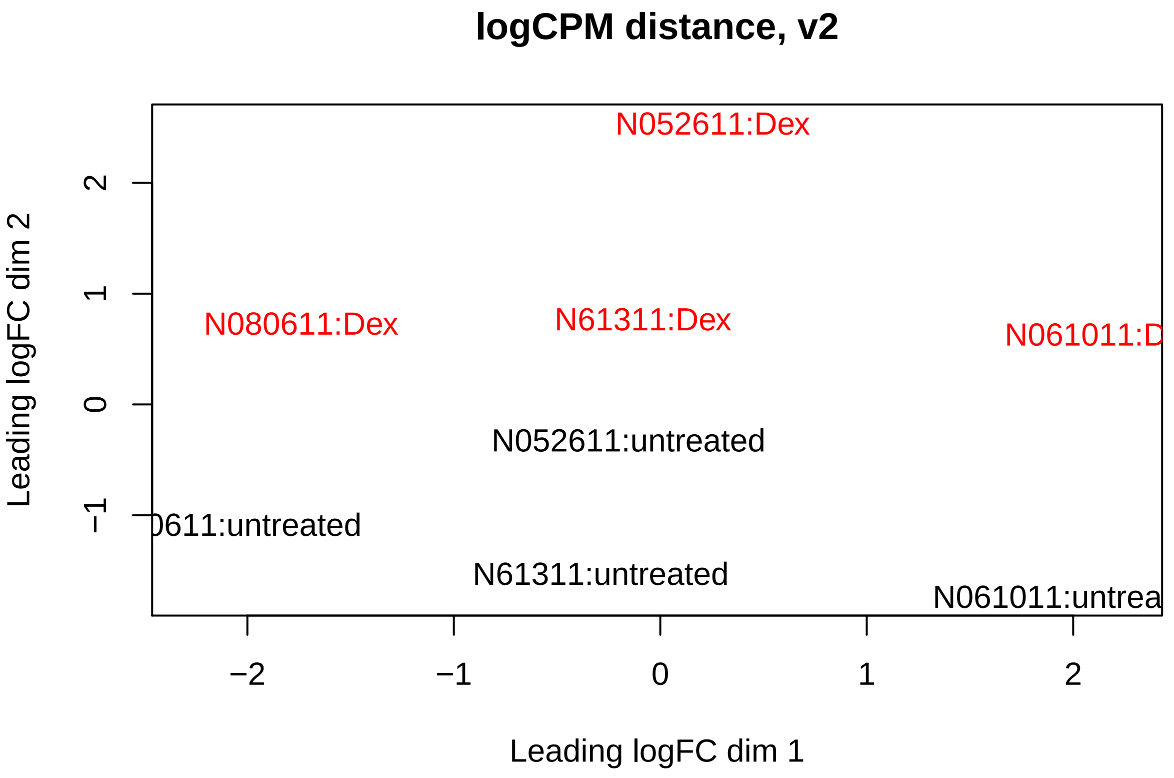
```
plotMDS(y2, main="BCV distance (filtered data), v2",
        col=c(rep(c("black","red"),4)),
        labels=paste(sampleTable$cells, sampleTable$treatment, sep=":")
          )
```



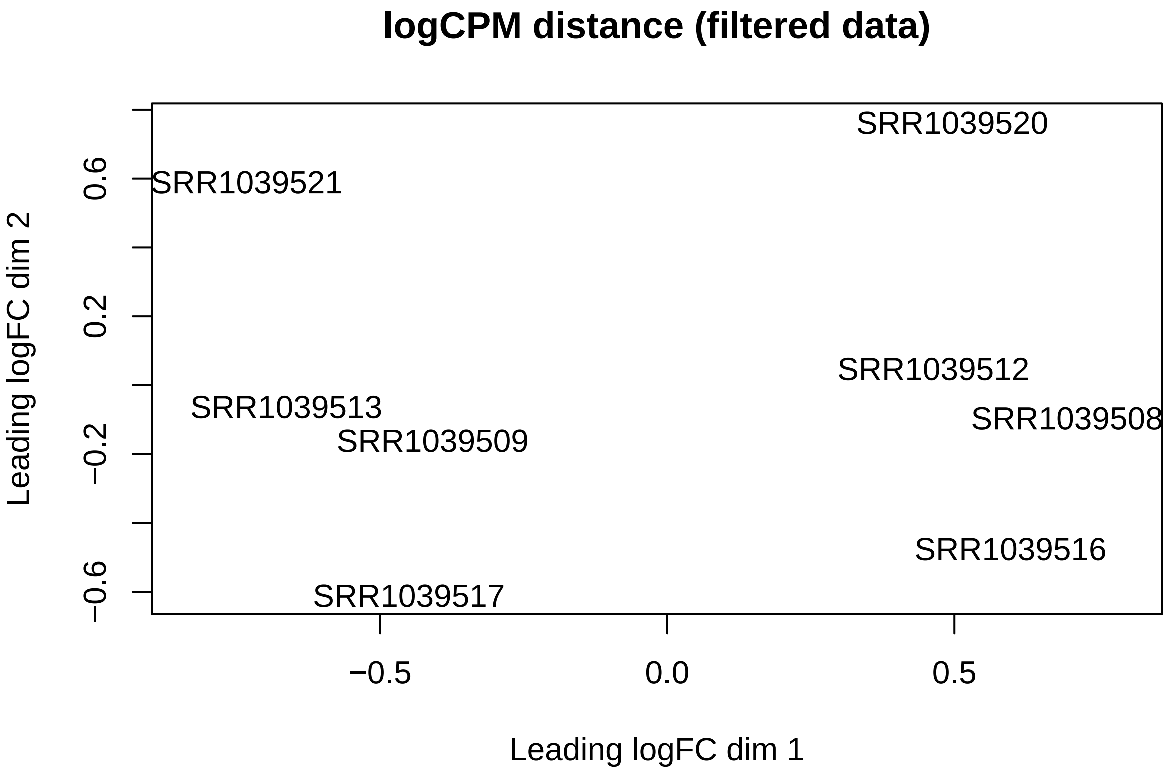
```
# log-transformation helps shrinking the data in the low range
# MDS plots that shows distances, in terms of shrunk 'cpm' changes
logCPM <- cpm(y, log=TRUE)
plotMDS(logCPM, main="logCPM distance")
```



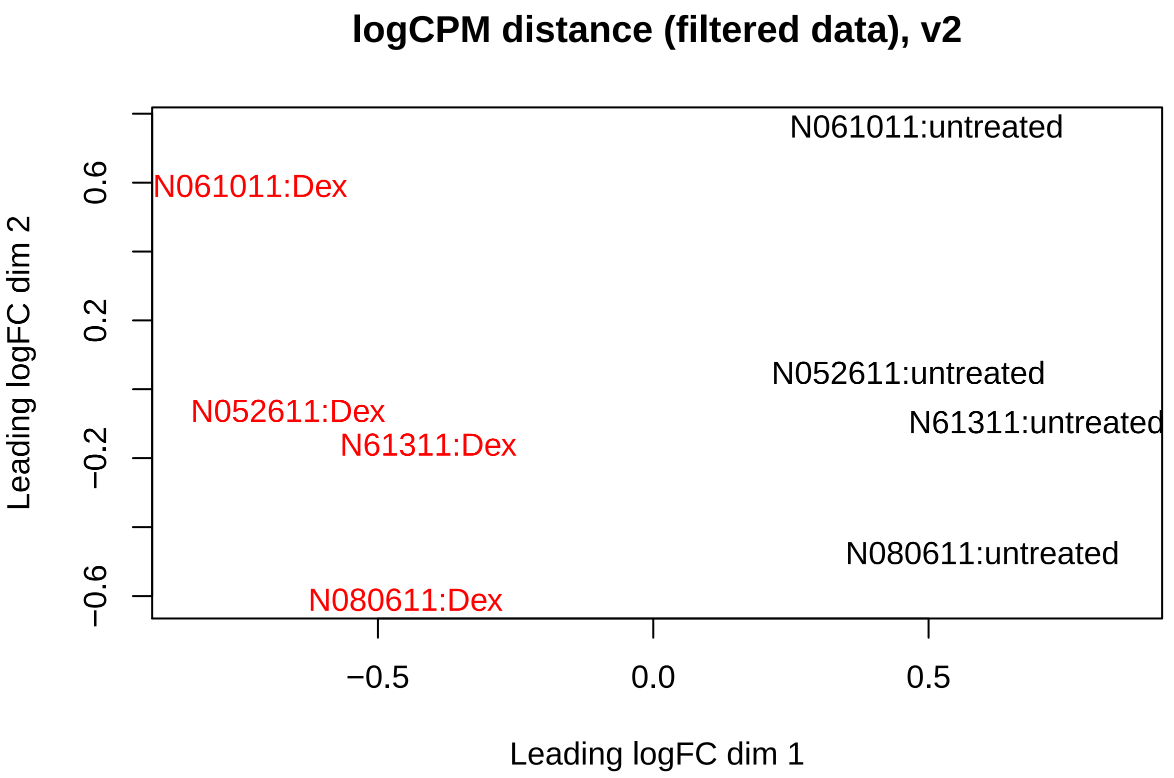
```
plotMDS(logCPM, main="logCPM distance, v2",
        col=c(rep(c("black","red"),4)),
        labels=paste(sampleTable$cells, sampleTable$treatment, sep=":")
        )
```



```
logCPM2 <- cpm(y2, log=TRUE)
plotMDS(logCPM2, main="logCPM distance (filtered data)")
```

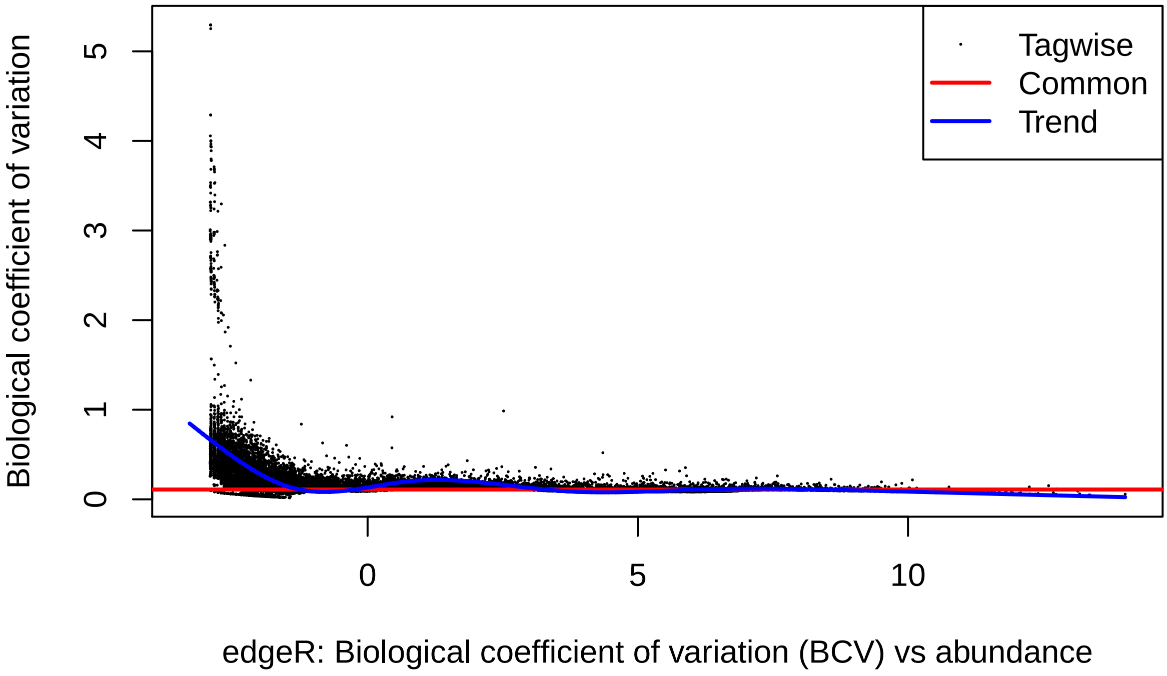


```
plotMDS(logCPM2, main="logCPM distance (filtered data), v2",
        col=c(rep(c("black","red"),4)),
        labels=paste(sampleTable$cells, sampleTable$treatment, sep=":")
        )
```

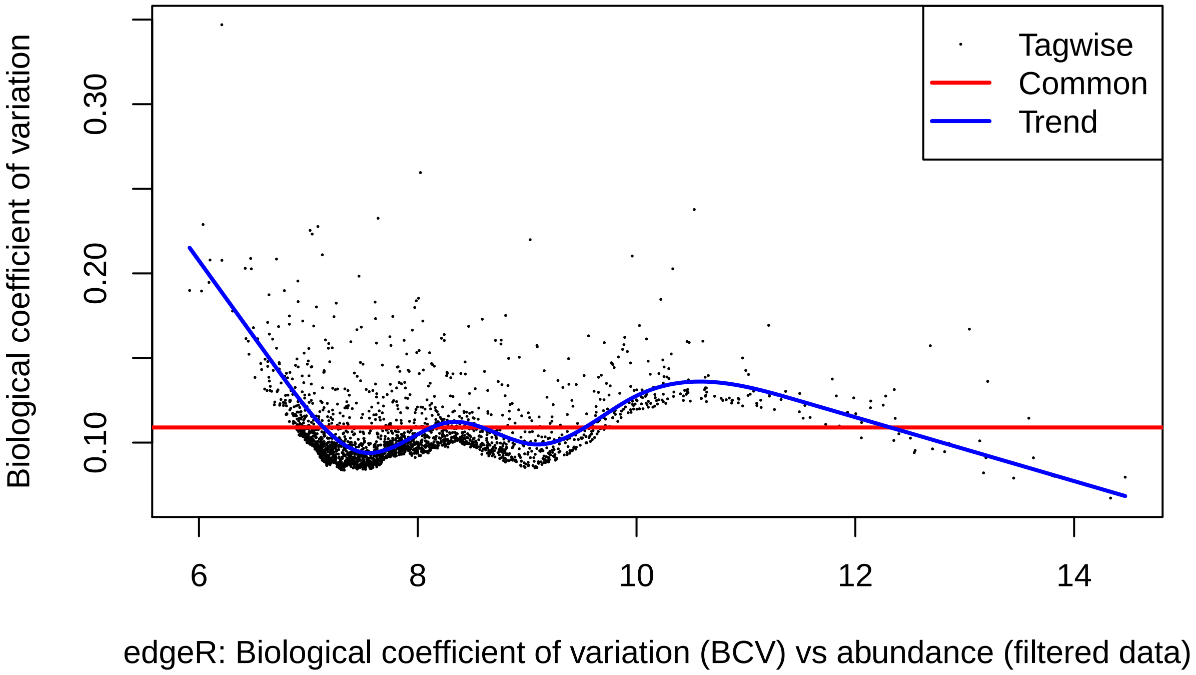


Biological coefficient of variation plot

```
plotBCV(y,
        "edgeR: Biological coefficient of variation (BCV) vs abundance")
```



```r
plotBCV(y2,
        "edgeR: Biological coefficient of variation (BCV) vs abundance (filtered data)")
```



The curly plot obtained on fitered data is not so ''normal''

build a GLM fitting model from the data

DE analysisi occurs in two steps:

- The function **glmFit()** fits the negative binomial GLM for each tag and produces an object of class DGEGLM with some new components.
- This DGEGLM object can then be passed to the function **glmLRT()** to carry out the likelihood ratio test.

```
fit <- glmFit(y, design)
fit2 <- glmFit(y2, design)

# apply fit to the data for Dex vs Untreated
lrt <- glmLRT(fit)
lrt2 <- glmLRT(fit2)

# review top findings
topTags(lrt)
```

```
## Coefficient:  treatmentDex
##                     logFC   logCPM        LR        PValue           FDR
## ENSG00000109906  7.330497 4.209885 1314.6160 7.533061e-288 4.352076e-283
## ENSG00000165995  3.420225 4.710180  815.1899 2.688670e-179 7.766628e-175
## ENSG00000152583  4.593225 5.656511  703.0387 6.529983e-155 1.257522e-150
## ENSG00000171819  5.809612 3.627514  647.4901 7.851890e-143 1.134068e-138
## ENSG00000163884  4.451690 4.806375  587.1334 1.053024e-129 1.216727e-125
## ENSG00000101347  3.745477 9.432151  578.3246 8.679752e-128 8.357589e-124
## ENSG00000189221  3.345965 6.874666  537.5559 6.414091e-119 5.293732e-115
## ENSG00000162692 -3.726771 4.760297  522.4378 1.247843e-115 9.011456e-112
## ENSG00000120129  2.935428 7.407823  493.0609 3.074725e-109 1.973734e-105
## ENSG00000127954  5.211009 4.241560  491.3005 7.427587e-109 4.291140e-105
```

```
# before filtering:
# Coefficient:  treatmentDex
#                     logFC   logCPM        LR        PValue           FDR
# ENSG00000109906  7.330497 4.209885 1314.6167 7.530502e-288 4.350597e-283
# ENSG00000165995  3.420225 4.710180  815.1903 2.688130e-179 7.765066e-175
# ENSG00000152583  4.593225 5.656511  703.0385 6.530830e-155 1.257686e-150
# ENSG00000171819  5.809612 3.627514  647.4904 7.850693e-143 1.133895e-138
# ENSG00000163884  4.451690 4.806375  587.1334 1.052997e-129 1.216696e-125
# ENSG00000101347  3.745477 9.432151  578.3239 8.682964e-128 8.360681e-124
# ENSG00000189221  3.345965 6.874666  537.5546 6.418499e-119 5.297371e-115
# ENSG00000162692 -3.726771 4.760297  522.4380 1.247757e-115 9.010832e-112
# ENSG00000120129  2.935428 7.407823  493.0571 3.080534e-109 1.977463e-105
# ENSG00000127954  5.211009 4.241560  491.3008 7.426448e-109 4.290482e-105

topTags(lrt2)
```

```
## Coefficient:  treatmentDex
##                     logFC   logCPM       LR        PValue           FDR
## ENSG00000189221  3.268833 7.317960 604.7934 1.517732e-133 3.302585e-130
## ENSG00000120129  2.859060 7.850420 528.0597 7.465593e-117 8.122566e-114
## ENSG00000101347  3.669226 9.872310 427.9689  4.497424e-95  3.262131e-92
## ENSG00000178695 -2.625352 7.619877 412.2671  1.176629e-91  6.400862e-89
## ENSG00000211445  3.623321 9.705215 397.3160  2.114564e-88  9.202581e-86
## ENSG00000157214  1.882460 7.675423 353.4079  7.673985e-79  2.783098e-76
## ENSG00000134243  2.092546 8.571024 334.2542  1.138243e-74  3.538310e-72
## ENSG00000152583  4.524233 6.100943 316.0063  1.073555e-70  2.920069e-68
## ENSG00000198624  2.833797 7.143985 314.7766  1.989255e-70  4.809577e-68
## ENSG00000125148  2.035365 7.045554 305.8739  1.730167e-68  3.764844e-66
```

```
# Coefficient:  treatmentDex
#                      logFC   logCPM       LR        PValue           FDR
# ENSG00000189221  3.268833 7.317960 604.7934 1.517732e-133 3.302585e-130
# ENSG00000120129  2.859060 7.850420 528.0597 7.465593e-117 8.122565e-114
# ENSG00000101347  3.669226 9.872310 427.9689  4.497424e-95  3.262131e-92
# ENSG00000178695 -2.625352 7.619877 412.2671  1.176629e-91  6.400861e-89
# ENSG00000211445  3.623321 9.705215 397.3160  2.114563e-88  9.202578e-86
# ENSG00000157214  1.882460 7.675423 353.4079  7.673983e-79  2.783098e-76
# ENSG00000134243  2.092546 8.571024 334.2542  1.138244e-74  3.538312e-72
# ENSG00000152583  4.524233 6.100943 316.0063  1.073556e-70  2.920072e-68
# ENSG00000198624  2.833797 7.143985 314.7766  1.989255e-70  4.809577e-68
# ENSG00000125148  2.035365 7.045554 305.8739  1.730167e-68  3.764844e-66
```

NOTE: The filtering effect is quite impressive and genes that were apparently very significant DE are now gone.

Compute differential expression

```
# count UR, DR and nouncchanged genes
summary(de <- decideTestsDGE(lrt))
```

```
##    [,1]
## -1  2107
## 0  53540
## 1   2126
```

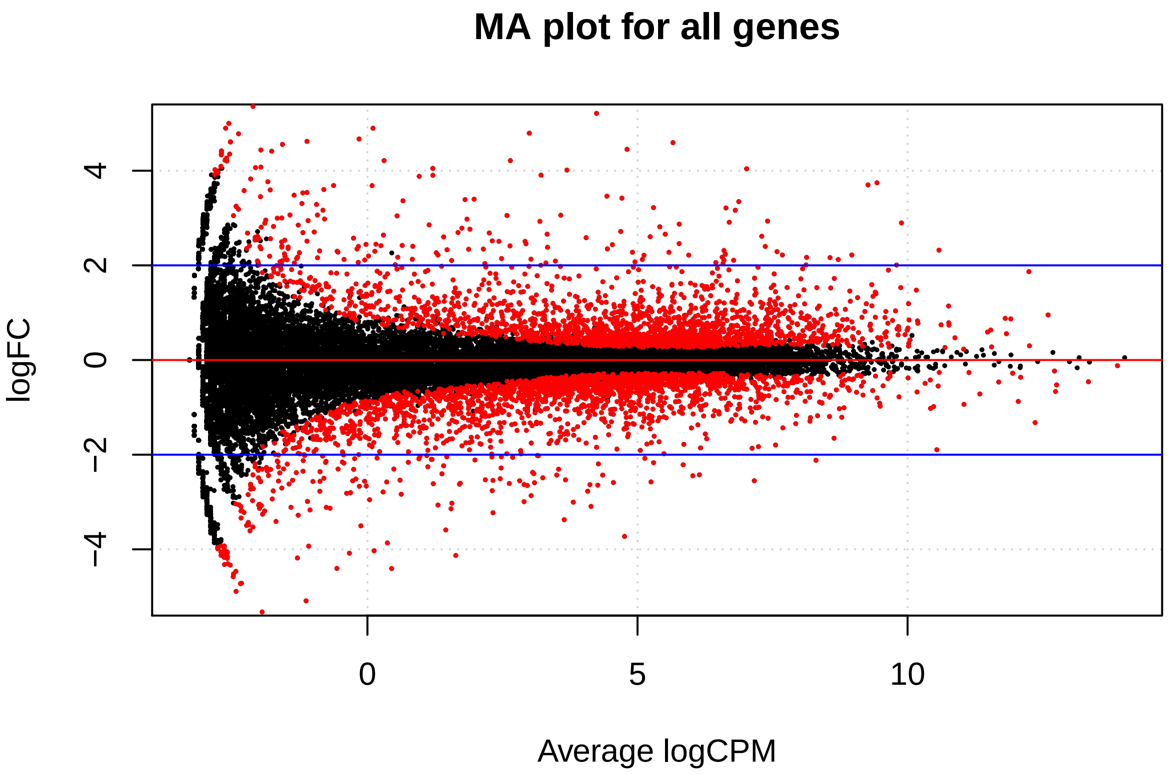
```
#    [,1]
# -1  2107
# 0  53540
# 1   2126

summary(de2 <- decideTestsDGE(lrt2))
```

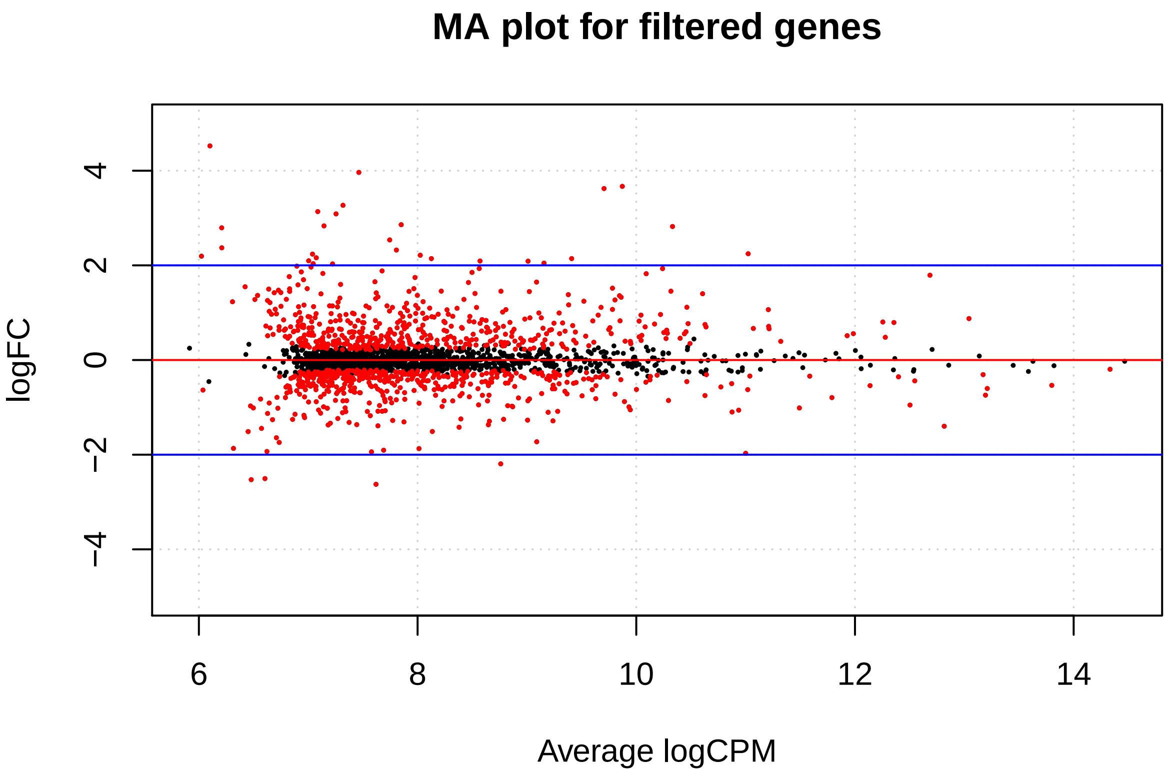
```
##    [,1]
## -1  479
## 0  1195
## 1   502
```

```
#    [,1]
# -1  479
# 0  1195
# 1   502

# MA-plot
detags <- rownames(y)[as.logical(de)]
plotSmear(lrt, de.tags=detags,
          ylim=c(-5,5),
          main="MA plot for all genes"
          )
abline(h = 0, col = "red")
abline(h = c(-2, 2), col = "blue")
```



```
# MA-plot on filtered data
detags2 <- rownames(y2)[as.logical(de2)]
plotSmear(lrt2, de.tags=detags2,
          ylim=c(-5,5),
          main="MA plot for filtered genes"
          )
abline(h = 0, col = "red")
abline(h = c(-2, 2), col = "blue")
```



Exploring and exporting results

```
# unfiltered data
results <- lrt$table

# add FDR to the table
results$padj <- p.adjust(results$PValue, method="BH")

# reorder by adjusted pvalue
results <- results[order(results$padj),]

# review top
head(results)
```

```
##                    logFC   logCPM        LR        PValue          padj
## ENSG00000109906 7.330497 4.209885 1314.6160 7.533061e-288 4.352076e-283
## ENSG00000165995 3.420225 4.710180  815.1899 2.688670e-179 7.766628e-175
## ENSG00000152583 4.593225 5.656511  703.0387 6.529983e-155 1.257522e-150
## ENSG00000171819 5.809612 3.627514  647.4901 7.851890e-143 1.134068e-138
## ENSG00000163884 4.451690 4.806375  587.1334 1.053024e-129 1.216727e-125
## ENSG00000101347 3.745477 9.432151  578.3246 8.679752e-128 8.357589e-124
```

```
#                    logFC   logCPM        LR        PValue          padj
# ENSG00000109906 7.330497 4.209885 1314.6167 7.530502e-288 4.350597e-283
# ENSG00000165995 3.420225 4.710180  815.1903 2.688130e-179 7.765066e-175
# ENSG00000152583 4.593225 5.656511  703.0385 6.530830e-155 1.257686e-150
# ENSG00000171819 5.809612 3.627514  647.4904 7.850693e-143 1.133895e-138
# ENSG00000163884 4.451690 4.806375  587.1334 1.052997e-129 1.216696e-125
# ENSG00000101347 3.745477 9.432151  578.3239 8.682964e-128 8.360681e-124

# save to file
write.csv(as.data.frame(results),
          file="EdgeR-Dex_vs_untreated_results.csv")

# filtered data
results2 <- lrt2$table

# add FDR to the table
results2$padj <- p.adjust(results2$PValue, method="BH")

# reorder by adjusted pvalue
results2 <- results2[order(results2$padj),]

# review top
head(results2)
```

```
##                      logFC   logCPM       LR        PValue          padj
## ENSG00000189221  3.268833 7.317960 604.7934 1.517732e-133 3.302585e-130
## ENSG00000120129  2.859060 7.850420 528.0597 7.465593e-117 8.122566e-114
## ENSG00000101347  3.669226 9.872310 427.9689  4.497424e-95  3.262131e-92
## ENSG00000178695 -2.625352 7.619877 412.2671  1.176629e-91  6.400862e-89
## ENSG00000211445  3.623321 9.705215 397.3160  2.114564e-88  9.202581e-86
## ENSG00000157214  1.882460 7.675423 353.4079  7.673985e-79  2.783098e-76
```

```
#                      logFC   logCPM       LR        PValue          padj
# ENSG00000189221  3.268833 7.317960 604.7934 1.517732e-133 3.302585e-130
# ENSG00000120129  2.859060 7.850420 528.0597 7.465593e-117 8.122565e-114
# ENSG00000101347  3.669226 9.872310 427.9689  4.497424e-95  3.262131e-92
# ENSG00000178695 -2.625352 7.619877 412.2671  1.176629e-91  6.400861e-89
# ENSG00000211445  3.623321 9.705215 397.3160  2.114563e-88  9.202578e-86
# ENSG00000157214  1.882460 7.675423 353.4079  7.673983e-79  2.783098e-76

# save to file
write.csv(as.data.frame(results2),
          file="EdgeR_filtered-Dex_vs_untreated_results.csv")
```

```
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] splines   parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
## [1] gplots_2.16.0       RColorBrewer_1.1-2  vsn_3.34.0
## [4] Biobase_2.26.0      BiocGenerics_0.12.1 ggplot2_1.0.1
## [7] edgeR_3.8.6         limma_3.22.7
##
## loaded via a namespace (and not attached):
##  [1] affy_1.44.0           affyio_1.34.0         BiocInstaller_1.16.2
##  [4] bitops_1.0-6          caTools_1.17.1        colorspace_1.2-6
##  [7] digest_0.6.8          evaluate_0.5.5        formatR_1.1
## [10] gdata_2.13.3          grid_3.1.2            gtable_0.1.2
## [13] gtools_3.4.1          htmltools_0.2.6       KernSmooth_2.23-14
## [16] knitr_1.9             lattice_0.20-31       MASS_7.3-40
## [19] munsell_0.4.2         plyr_1.8.1            preprocessCore_1.28.0
## [22] proto_0.3-10          Rcpp_0.11.5           reshape2_1.4.1
## [25] rmarkdown_0.5.1       scales_0.2.4          stringr_0.6.2
## [28] tools_3.1.2           yaml_2.1.13           zlibbioc_1.12.0
```

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