DESeq2 Workflow

Asad

4/18/2023

1. Calling libraries

```
library(DESeq2)
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

```
## Loading required package: IRanges
```

```
##
## Attaching package: 'IRanges'
```

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

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

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

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

```
##
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:AnnotationDbi':
##
##     select

## The following object is masked from 'package:Biobase':
##
##     combine

## The following object is masked from 'package:matrixStats':
##
##     count

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
library(EnhancedVolcano)
```

```
## Loading required package: ggrepel

## Registered S3 methods overwritten by 'ggalt':
##   method                  from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob  ggplot2
##   grobX.absoluteGrob      ggplot2
##   grobY.absoluteGrob      ggplot2
```

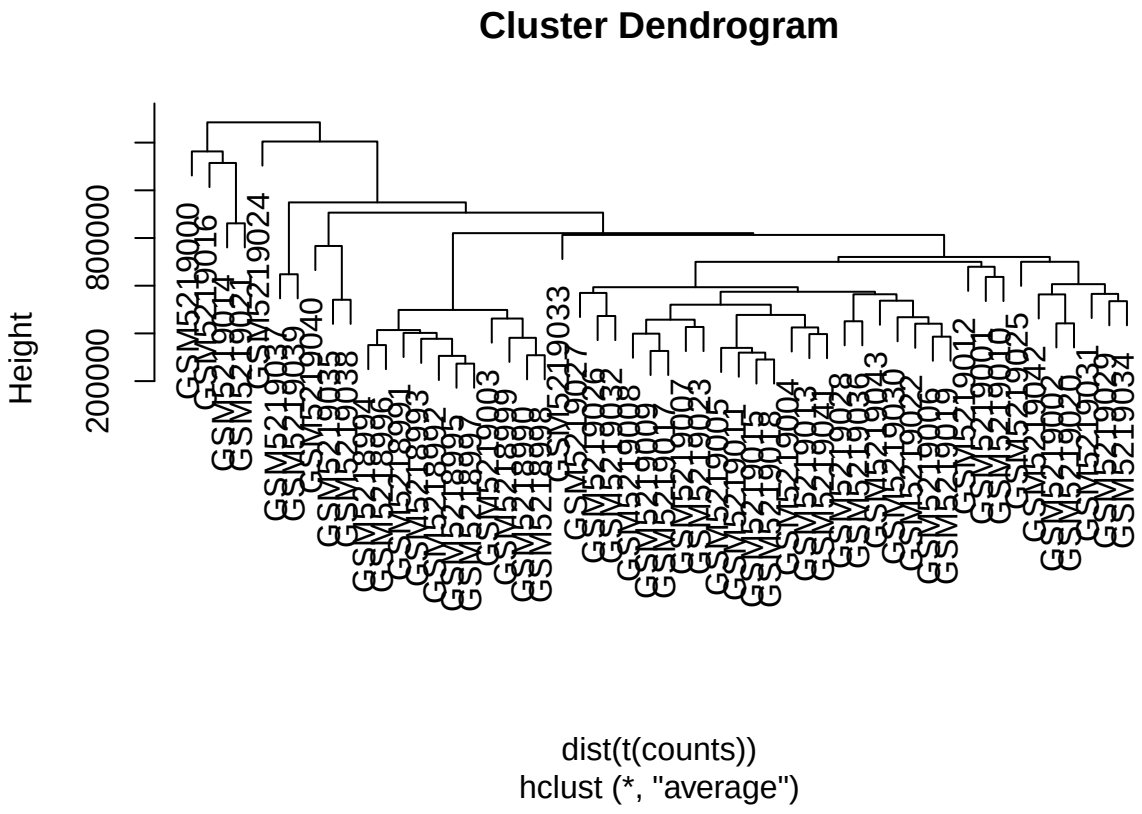
```
library(textshaping)
library(magrittr)
```

2. Reading Counts and Sample Information

```
counts <- read.csv('D:/GSE171110_COVID/GSE171110_CovidBlood.csv', header = T, row.names = 1)
info <- read.csv('D:/GSE171110_COVID/sample_info.csv', header = T, row.names = 1)
```

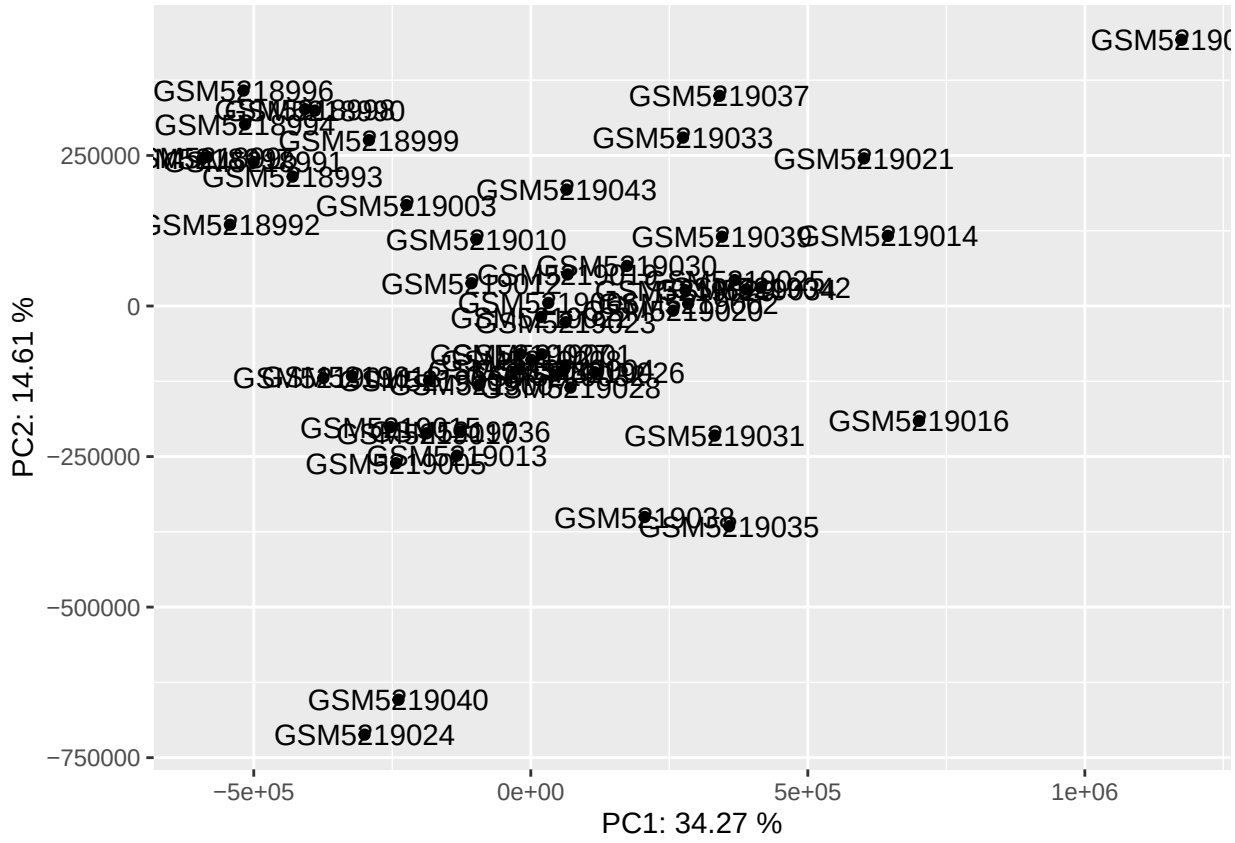
3. #Identifying outliers #Method 1 with hierarchical clustering

```
htree<- hclust(dist(t(counts)), method = 'average')
plot(htree)
```



#Method 2 with PCA Analysis

```
pca <- prcomp(t(counts))
pca.dat <- pca$x

pca.var <- pca$sdev^2
pca.var.percent <- round(pca.var/sum(pca.var)*100, digits = 2)

pca.dat <- as.data.frame(pca.dat)

ggplot(pca.dat, aes(PC1, PC2)) +
  geom_point() +
  geom_text(label = rownames(pca.dat)) +
  labs(x = paste0('PC1: ', pca.var.percent[1], ' %'),
       y = paste0('PC2: ', pca.var.percent[2], ' %'))
```



4. Removing outliers

```
samples.to.be.excluded <- c('GSM5219000', 'GSM5219016', 'GSM5219014',
                            'GSM5219021', 'GSM5219024', 'GSM5219040',
                            'GSM5219000','GSM5219037', 'GSM5219039',
                            'GSM5219012')

final_counts <- counts[,!(colnames(counts) %in% samples.to.be.excluded)]
final_info<- info[!(row.names(info) %in% samples.to.be.excluded),]
```

In this Step We Need to Collapse Technical Replicates and Take Sums if the dataset contains TR

#If we know the batch information then:

```
final_info$Batch=as.factor(final_info$Batch)
```

5. Method I (batch known): Setting condition and Removing low-expressed genes

```
dds <- DESeqDataSetFromMatrix(countData=final_counts,
      colData=final_info, design =  ~ Batch+Condition)
```

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

```
keep <- rowSums(counts(dds)) > 50
dds <- dds[keep,]
```

6. Method 2: In case the Batch information is not known

```
dds <- DESeqDataSetFromMatrix(final_counts, final_info, ~Condition)
```

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

```
keep <- rowSums(counts(dds)) > 50
dds <- dds[keep,]
```

7. Set reference groups

```
dds$Condition<-relevel(dds$Condition, ref = 'Control')
```

8. Main DESEQ

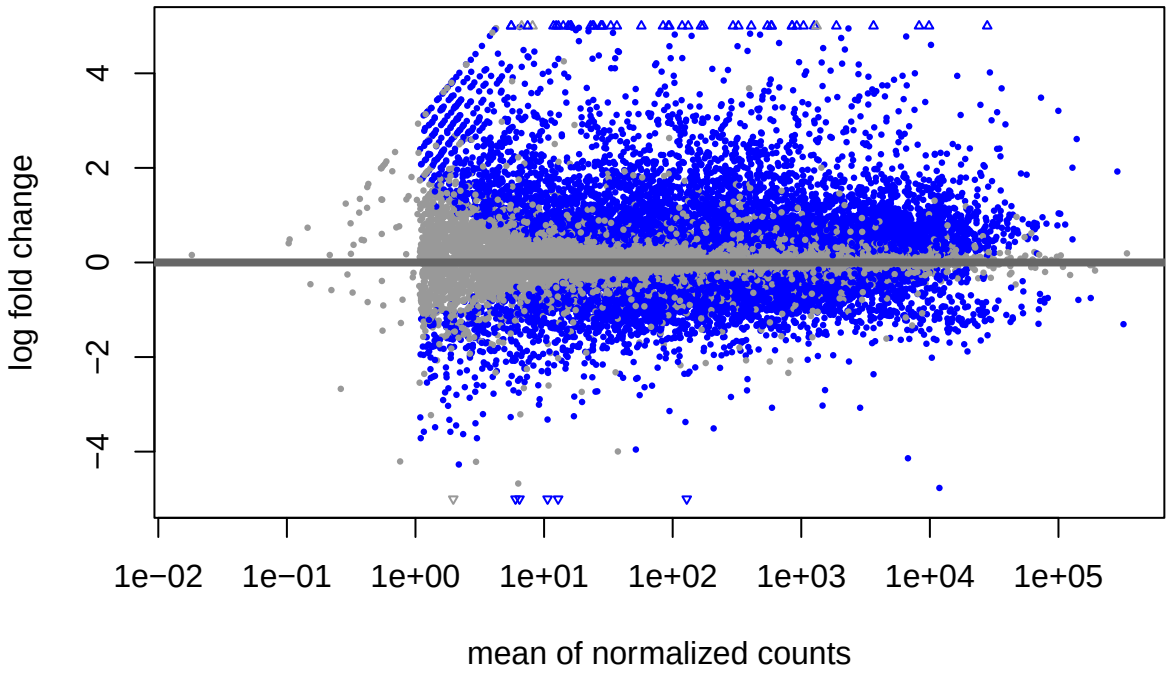
```
ddsDE <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 219 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)
```

```
## estimating dispersions

## fitting model and testing
```

9. MA Plot

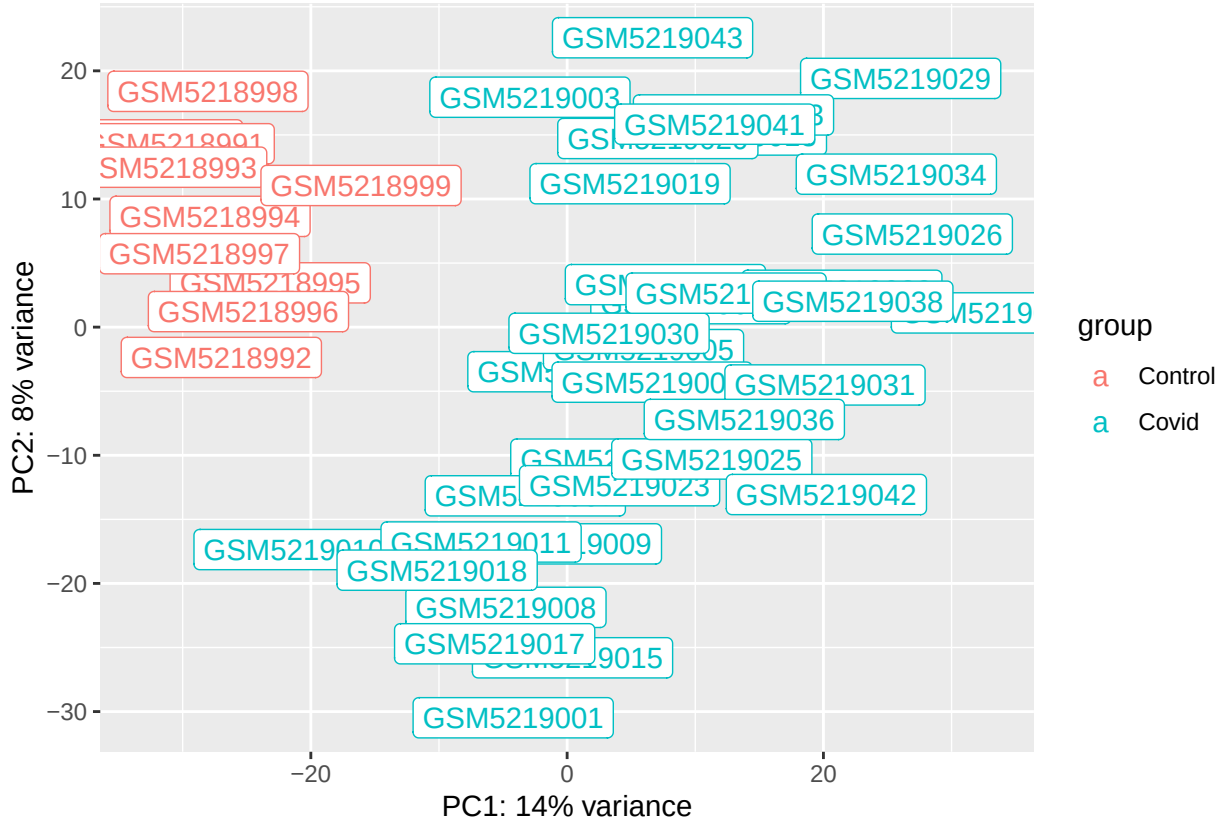
```
plotMA(ddsDE, ylim=c(-5,5))
```




```
vsdata <- vst(ddsDE, blind = FALSE)
```

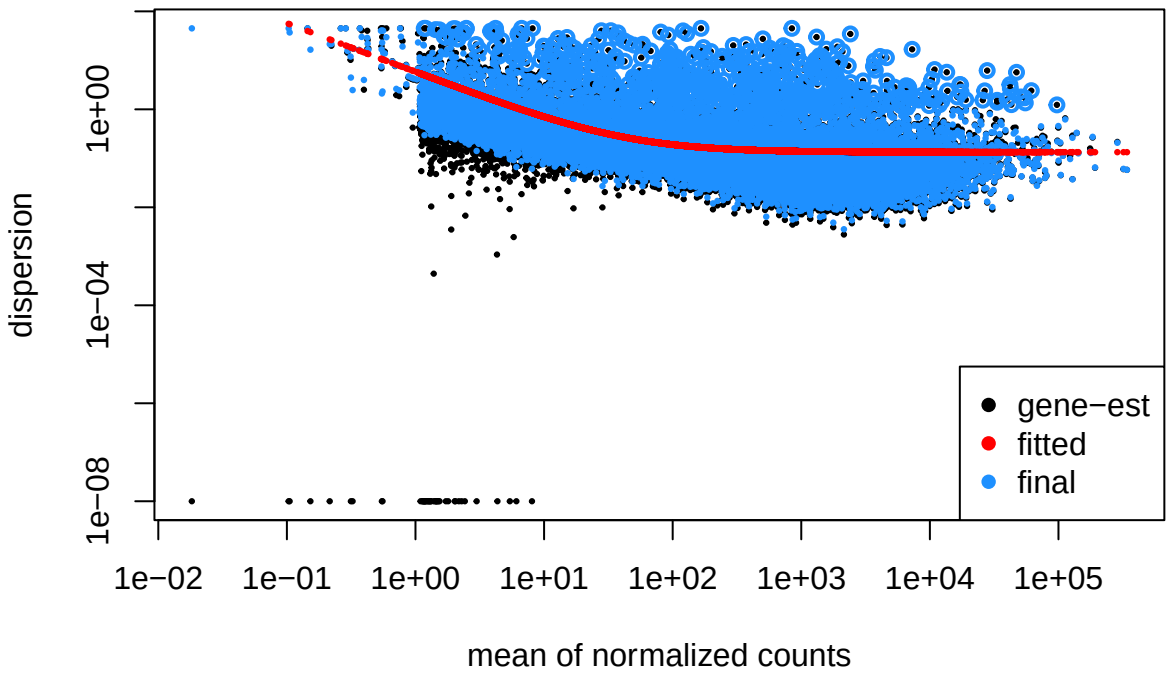
10. PCA Plot

```
plotPCA(vsdata, intgroup= "Condition")+
  geom_label(aes(label = name))
```



11. Dispersion Plot

```
plotDispEsts(ddsDE)
```



12.Save Normalized Read Counts

```
normCounts <-counts(ddsDE, normalized=T)
write.csv(normCounts, "Normalized Read Counts.csv")
```

13. Retrieve DESeq Result

```
res<- results(ddsDE, alpha = 0.05)
write.csv(res, file = 'DEGs.csv')
```

14. Make Contrast between Design

```
resultsNames(ddsDE)
```

```
## [1] "Intercept"                  "Condition_Covid_vs_Control"
```

```
con.res<- results(ddsDE, contrast = c("Condition", 'Control', 'Covid'))
```

15. Check Summary of the Results

```
summary(res)
```

```
##
## out of 19238 with nonzero total read count
```

```
## adjusted p-value < 0.05
## LFC > 0 (up)       : 5198, 27%
## LFC < 0 (down)     : 4281, 22%
## outliers [1]       : 0, 0%
## low counts [2]     : 8, 0.042%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

16. Reorder Results

```
resOrdered <- res[order(res$padj),]
```

16. Convert ENSEMBL ID to Gene Symbol

```
resOrdered$symbol<- mapIds(org.Hs.eg.db, keys=rownames(resOrdered), keytype = "ENSEMBL", column = "SYMB
```

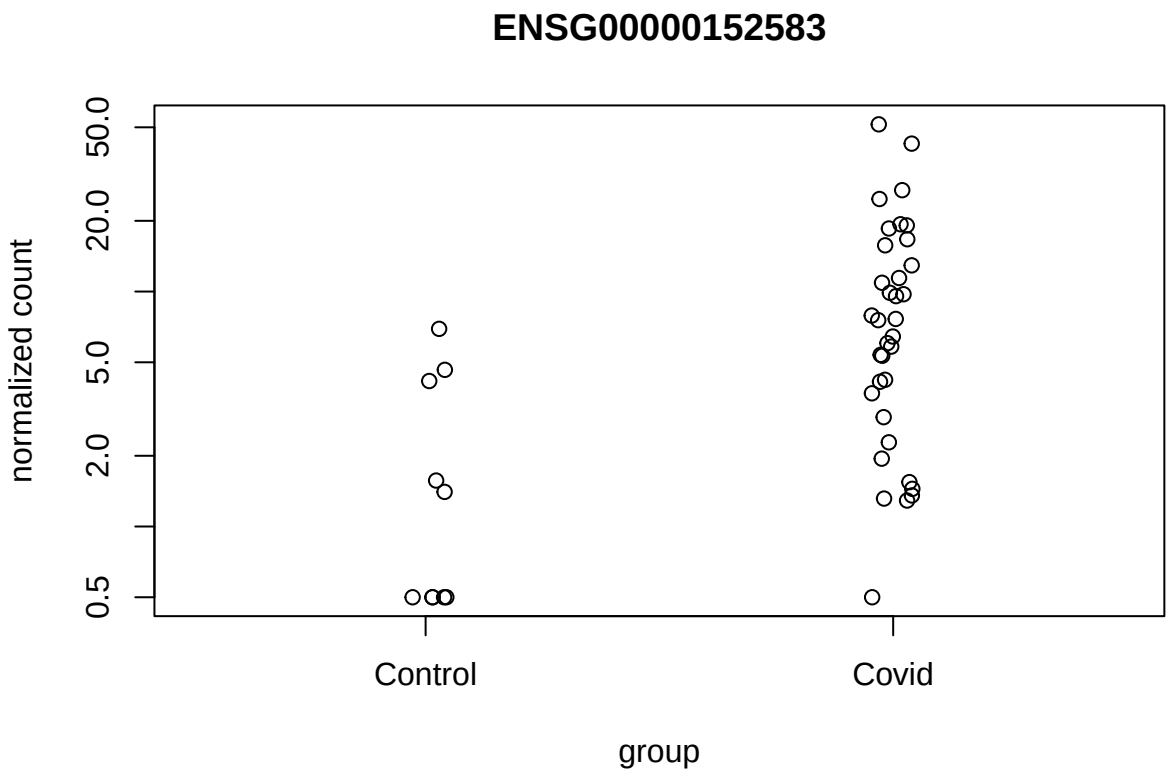
```
## 'select()' returned 1:many mapping between keys and columns
```

```
head(resOrdered)
```

```
## log2 fold change (MLE): Condition Covid vs Control
## Wald test p-value: Condition Covid vs Control
## DataFrame with 6 rows and 7 columns
##                  baseMean log2FoldChange     lfcSE      stat      pvalue
##                 <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000148773  3162.081        2.73968  0.200406   13.6706 1.52086e-42
## ENSG00000075218   237.971        2.82169  0.210982   13.3741 8.57211e-41
## ENSG00000186185   227.031        2.66959  0.200981   13.2828 2.91445e-40
## ENSG00000101057  1906.474        3.49777  0.265181   13.1901 1.00013e-39
## ENSG00000143603   319.357        4.57336  0.352181   12.9858 1.47280e-38
## ENSG00000109805   492.680        2.72956  0.211283   12.9190 3.51742e-38
##                       padj      symbol
##                  <numeric> <character>
## ENSG00000148773 2.92583e-38       MKI67
## ENSG00000075218 8.24551e-37       GTSE1
## ENSG00000186185 1.86894e-36      KIF18B
## ENSG00000101057 4.81014e-36       MYBL2
## ENSG00000143603 5.66673e-35       KCNN3
## ENSG00000109805 9.66688e-35       NCAPG
```

17. Plot a Single Transcript

```
plotCounts(ddsDE, gene="ENSG00000152583", intgroup="Condition")
```



18. Stratify

```
res1 <- as.data.frame(resOrdered)
res1<- filter(res1, padj<0.01, log2FoldChange>1 | log2FoldChange< -1)
```

19. Remove Duplicates ans Save

```
Significant_DEGs<- res1[!duplicated(res1$symbol),]

write.csv(Significant_DEGs, file = 'Significant_DEGs.csv')
```

20. Save Significant UPs and Downs

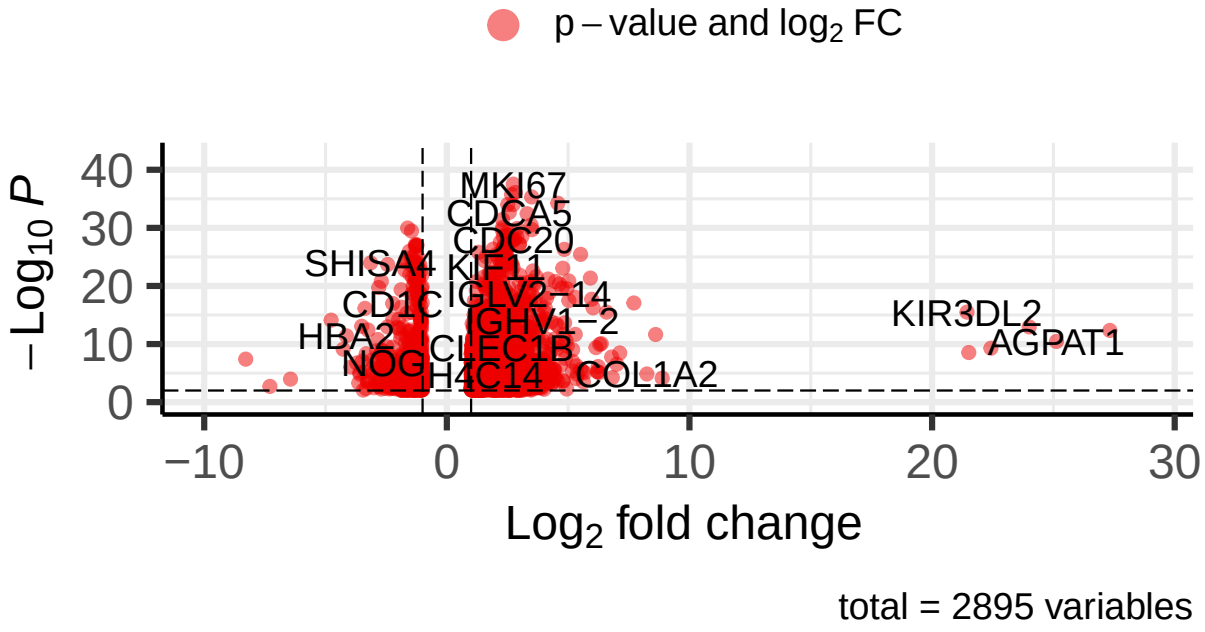
```
Significant_Up<- subset(Significant_DEGs,log2FoldChange>1)
Significant_Down<- subset(Significant_DEGs,log2FoldChange< -1)

write.csv(Significant_Up, file = 'Up_DEGs.csv')
write.csv(Significant_Down, file = 'Down_DEGs.csv')
```

21. Volcano Plot

```
#Method 1
Plot.df<-data.frame(res1)

EnhancedVolcano(Plot.df, x="log2FoldChange", y="padj", lab = Plot.df$symbol, pCutoff = 0.01, FCcutoff =
```

Volcano plot

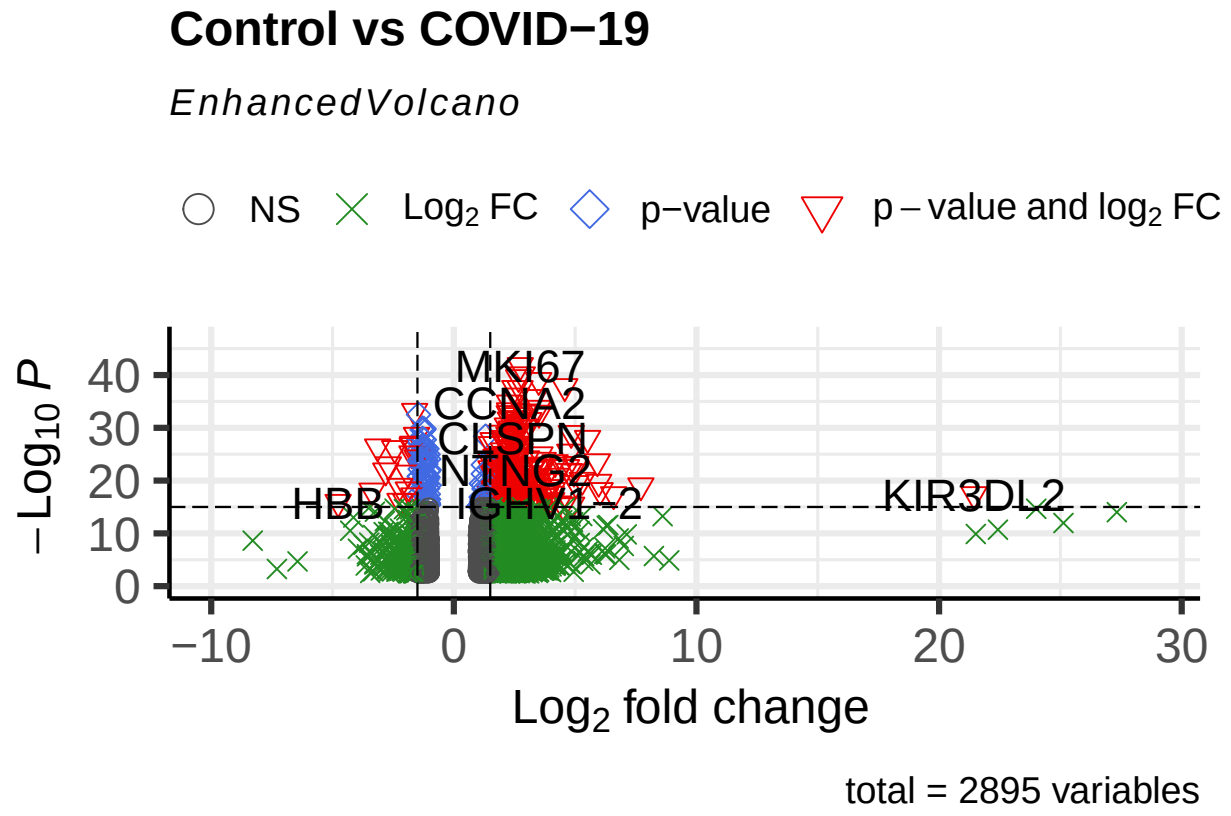
EnhancedVolcano



total = 2895 variables

```
#Method 2
EnhancedVolcano(Plot.df,
                lab = Plot.df$symbol,
                x = 'log2FoldChange',
                y = 'pvalue',
                title = 'Control vs COVID-19',
                pCutoff = 10e-16,
                FCcutoff = 1.5,
                pointSize = 3.0,
                labSize = 6.0,
                shape = c(1, 4, 23, 25),
                colAlpha = 1)
```

```
## Warning: The ‘guide‘ argument in ‘scale_*()‘ cannot be ‘FALSE‘. This was deprecated in
## ggplot2 3.3.4.
## i Please use "none" instead.
## i The deprecated feature was likely used in the EnhancedVolcano package.
##   Please report the issue to the authors.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

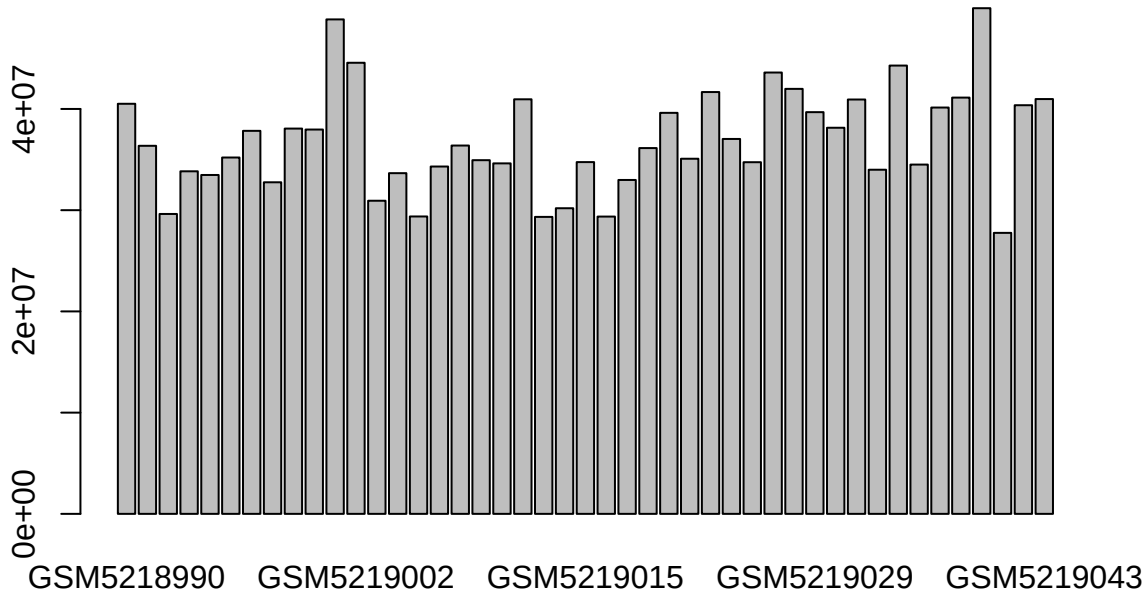


22. Quality Assessment

```
#Read counts distribution across samples
colSums(counts(dds))
```

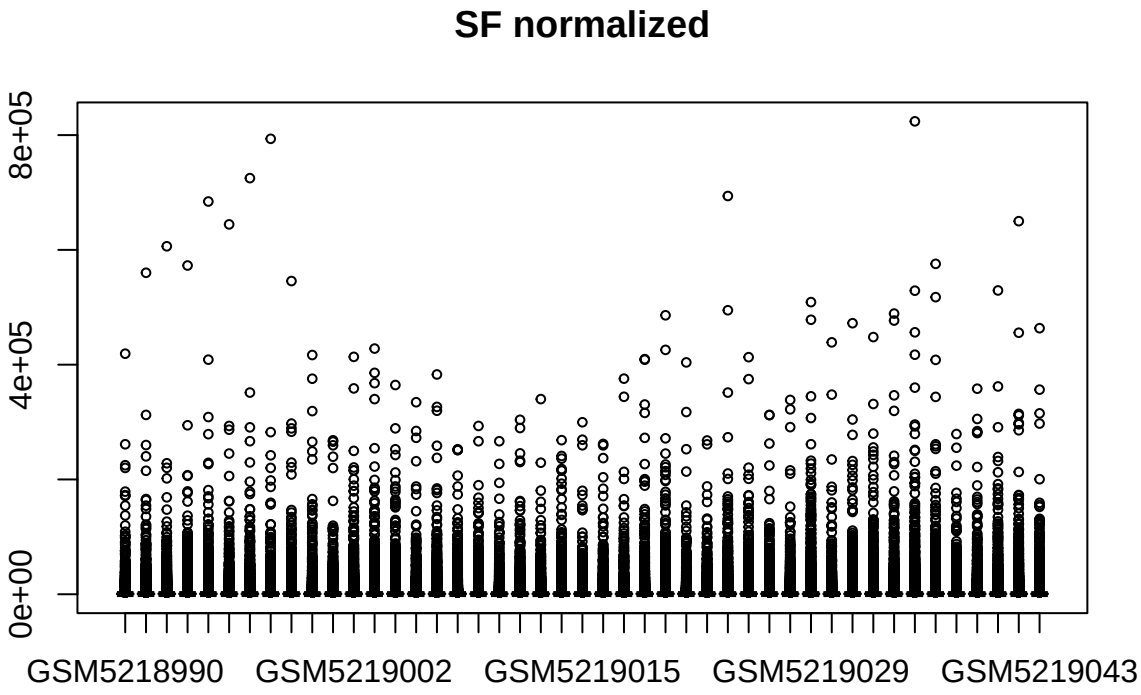
```
## GSM5218990 GSM5218991 GSM5218992 GSM5218993 GSM5218994 GSM5218995 GSM5218996
##   40510278   36357676   29624967   33841273   33476433   35201971   37836937
## GSM5218997 GSM5218998 GSM5218999 GSM5219001 GSM5219002 GSM5219003 GSM5219004
##   32756521   38059597   37965323   48846761   44562512   30932685   33655679
## GSM5219005 GSM5219006 GSM5219007 GSM5219008 GSM5219009 GSM5219010 GSM5219011
##   29378131   34312997   36384263   34938748   34623477   40953015   29336680
## GSM5219013 GSM5219015 GSM5219017 GSM5219018 GSM5219019 GSM5219020 GSM5219022
##   30191083   34749584   29366868   32983871   36134064   39614031   35081845
## GSM5219023 GSM5219025 GSM5219026 GSM5219027 GSM5219028 GSM5219029 GSM5219030
##   41668740   37036750   34738910   43597274   41981991   39682684   38142718
## GSM5219031 GSM5219032 GSM5219033 GSM5219034 GSM5219035 GSM5219036 GSM5219038
##   40932534   33994758   44284348   34505322   40140362   41122646   49953091
## GSM5219041 GSM5219042 GSM5219043
##   27758432   40368412   40976223
```

```r
lol<- colSums(counts(dds)) %>% barplot
```

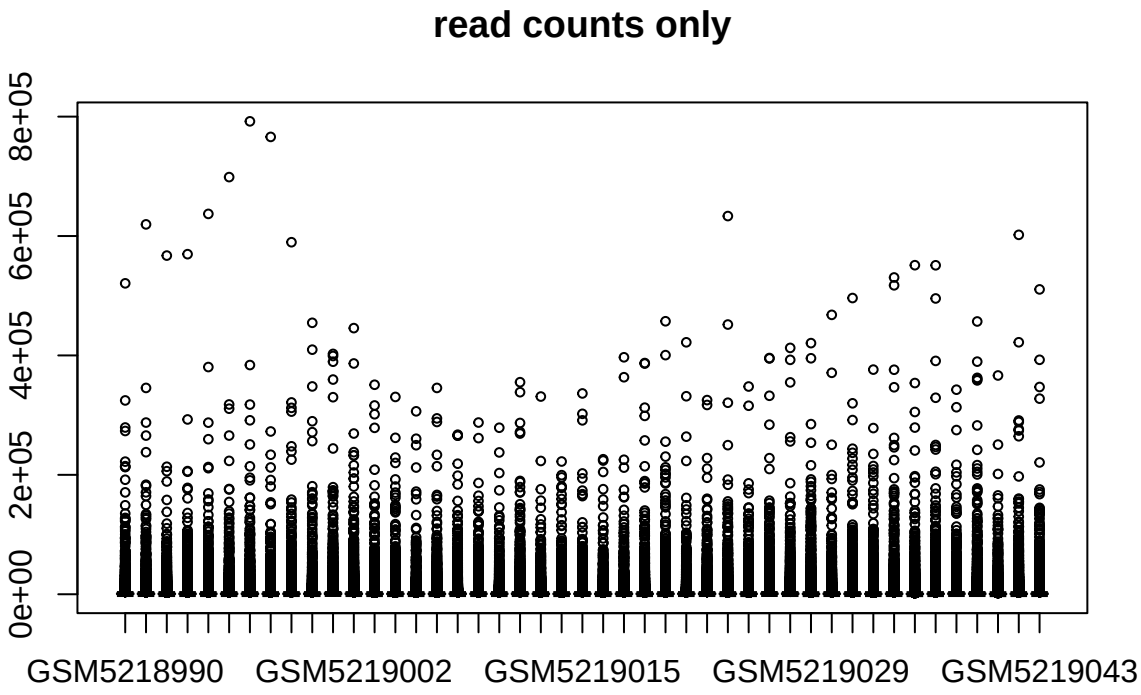



```r
par(mfrow=c(1,2))
```

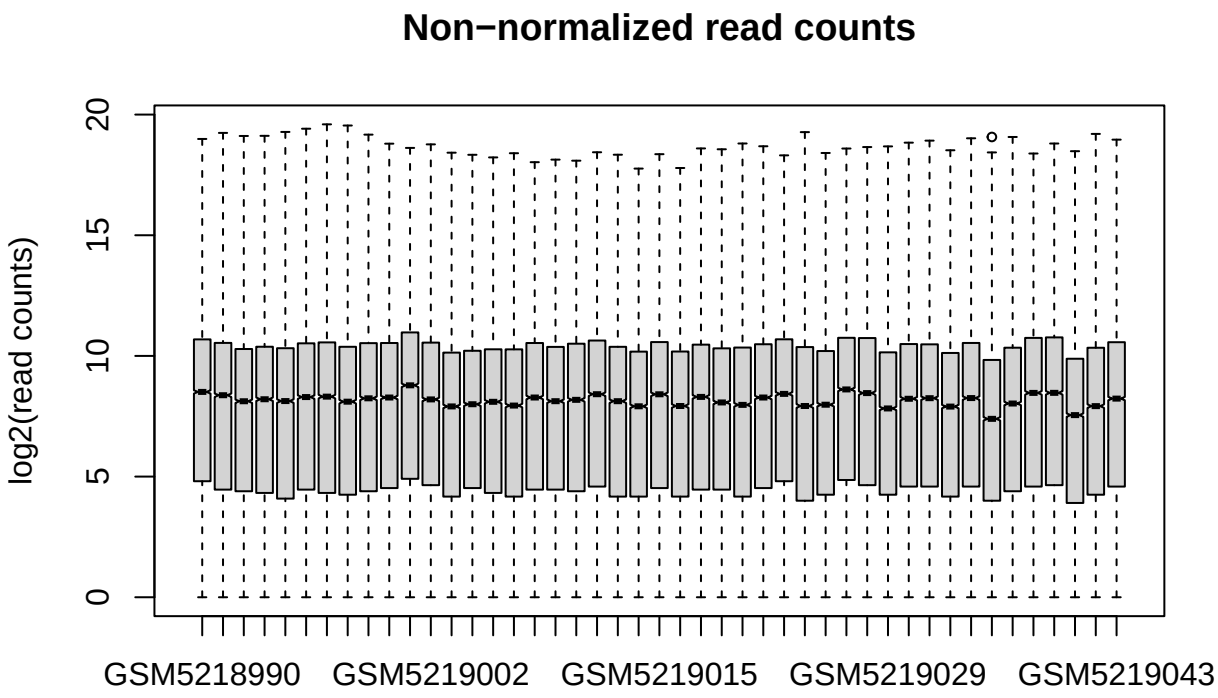
```r
#adding the boxplots
counts.sf_normalized <- counts(ddsDE, normalized=TRUE)
boxplot(counts.sf_normalized, main = "SF normalized", cex = .6)
```



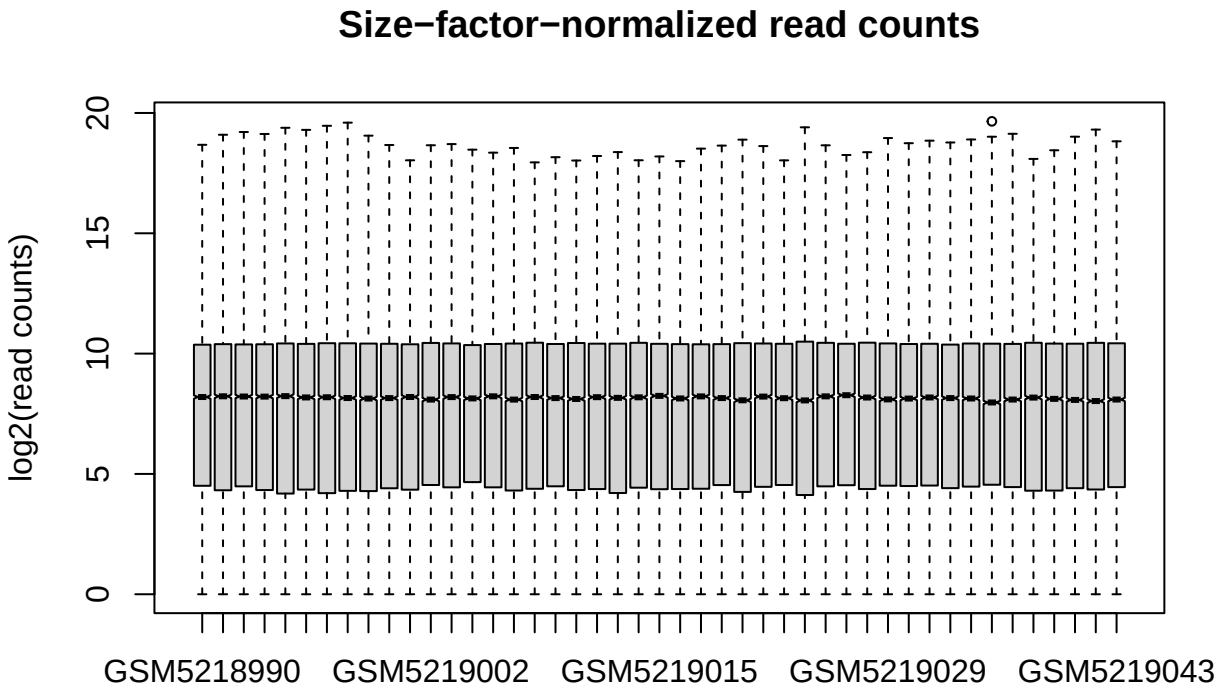

```
boxplot(counts(dds), main = "read counts only", cex = .6)
```




```
#Boxplots of read counts before and after normalization
boxplot(log2(counts(ddsDE)+1), notch=TRUE,
        main = "Non-normalized read counts",
        ylab="log2(read counts)", cex = .6)
```




```
boxplot(log2(counts(ddsDE, normalize= TRUE) +1), notch=TRUE,
        main = "Size-factor-normalized read counts",
        ylab="log2(read counts)", cex = .6)
```


Size−factor−normalized read counts
log2(read counts)
0
5
10
15
20
GSM5218990
GSM5219002
GSM5219015
GSM5219029
GSM5219043