DESeq2 Workflow

Asad

2023-05-15

```
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 object is masked from 'package:utils':
##
##     findMatches

## 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)
library(lazyeval)
library(reshape2)
library(gplots)
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:IRanges':
##
##     space
```

```
## The following object is masked from 'package:S4Vectors':
##
##     space
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(RColorBrewer)
setwd('E:/Differential Gene Expression/DESeq2 Workflow')
```

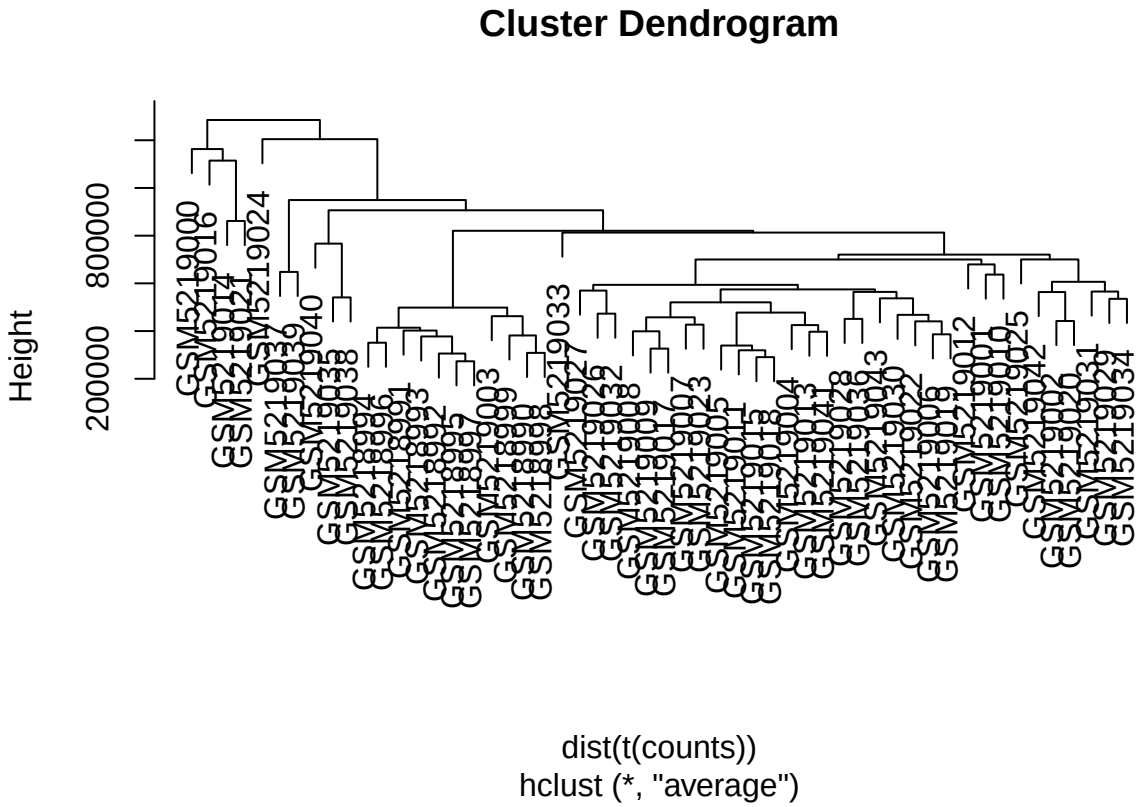
Reading Counts and Sample Information

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

Identifying outliers

#Method 1 with hierarchical clustering

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



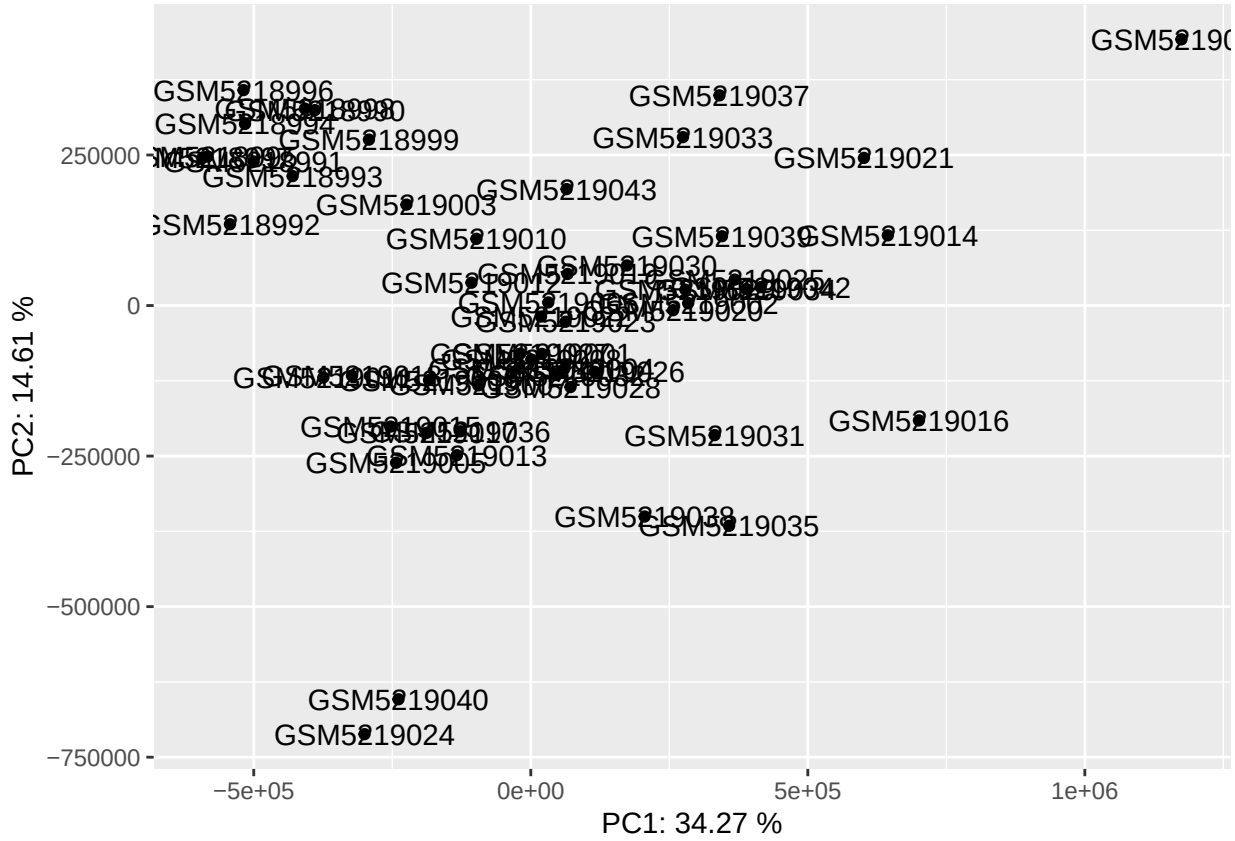
Method 2 with PCA Analysis

```
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], ' %'))
```



#Removing outliers

```
samples.to.be.excluded <- c('GSM5219000','GSM5219024', 'GSM5219040')
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

Examples for collapsing replicates

dds <- makeExampleDESeqDataSet(m=12) #make data with two technical replicates for three samples dds$sample <- factor(sample(paste0("sample", rep(1 : 9, c(2, 1, 1, 2, 1, 1, 2, 1, 1)))))dds$run <- paste0("run",1:12) ddsColl <- collapseReplicates(dds, dds$sample, dds$run) #examine the colData and column names of the collapsed data colData(ddsColl) colnames(ddsColl) #check DESeq2 user manual for more details

Fitting Design and filtering

```
final_info$Batch=as.factor(final_info$Batch)
final_info$Condition=as.factor(final_info$Condition)#Account for batch
dds <- DESeqDataSetFromMatrix(countData = final_counts,
                              colData=final_info, design = ~Condition) #+Batch
```

```
keep <- rowSums(counts(dds))>=10 #Recommended by user manual
dds <- dds[keep,]
dim(dds)
```

```
## [1] 21807    51
```

#Set reference groups

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

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 417 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

```
resultsNames(ddsDE)
```

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

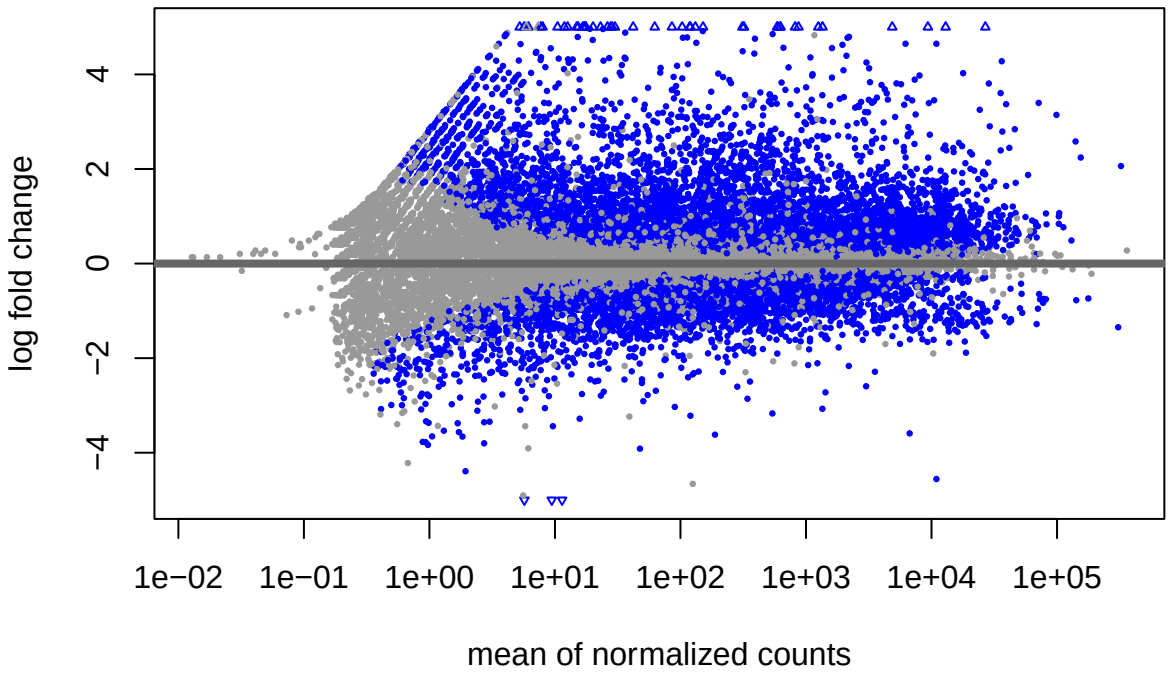
Save normalized counts

```
norm.counts<- counts(ddsDE, normalized=T)
log.norm.counts<- log2(counts(ddsDE, normalized=T))
write.csv(log.norm.counts, 'log.exprs.csv')
```

Quality assessment and visualization of the distribution of the data

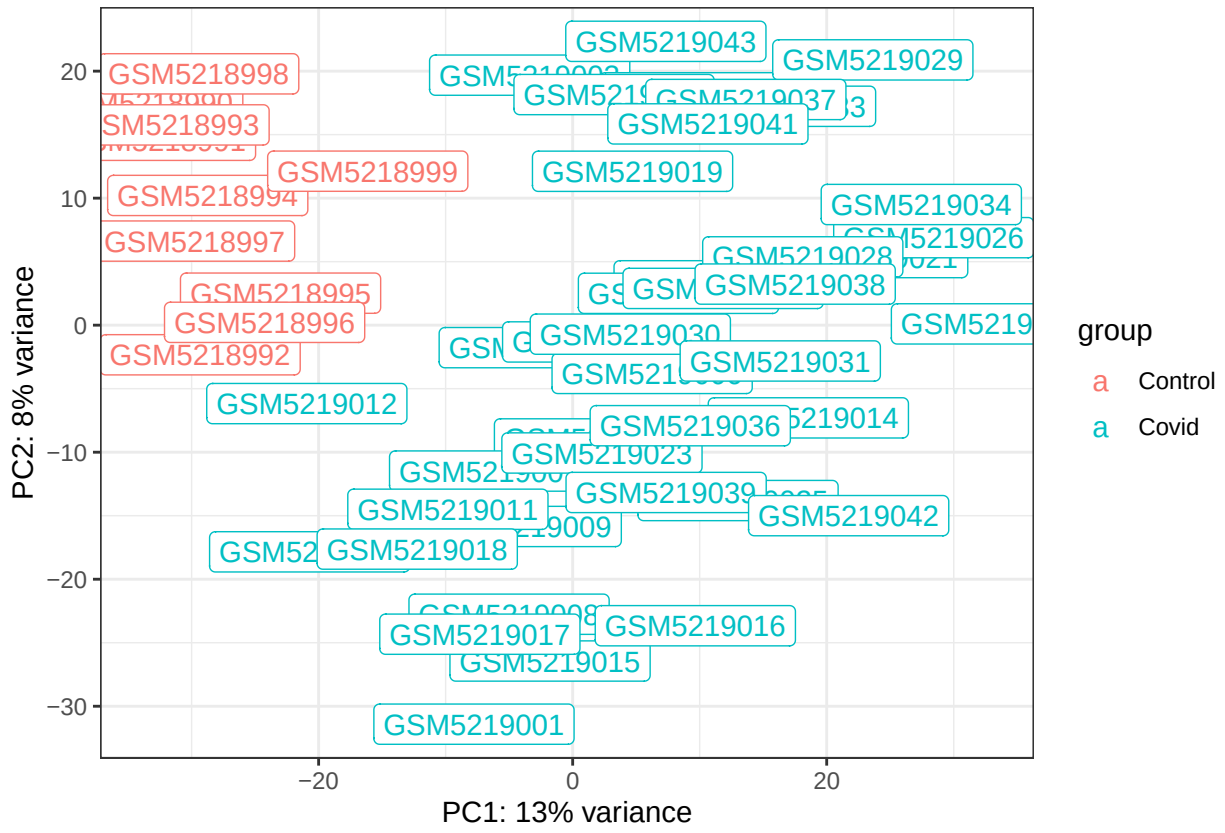
#MA plot

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



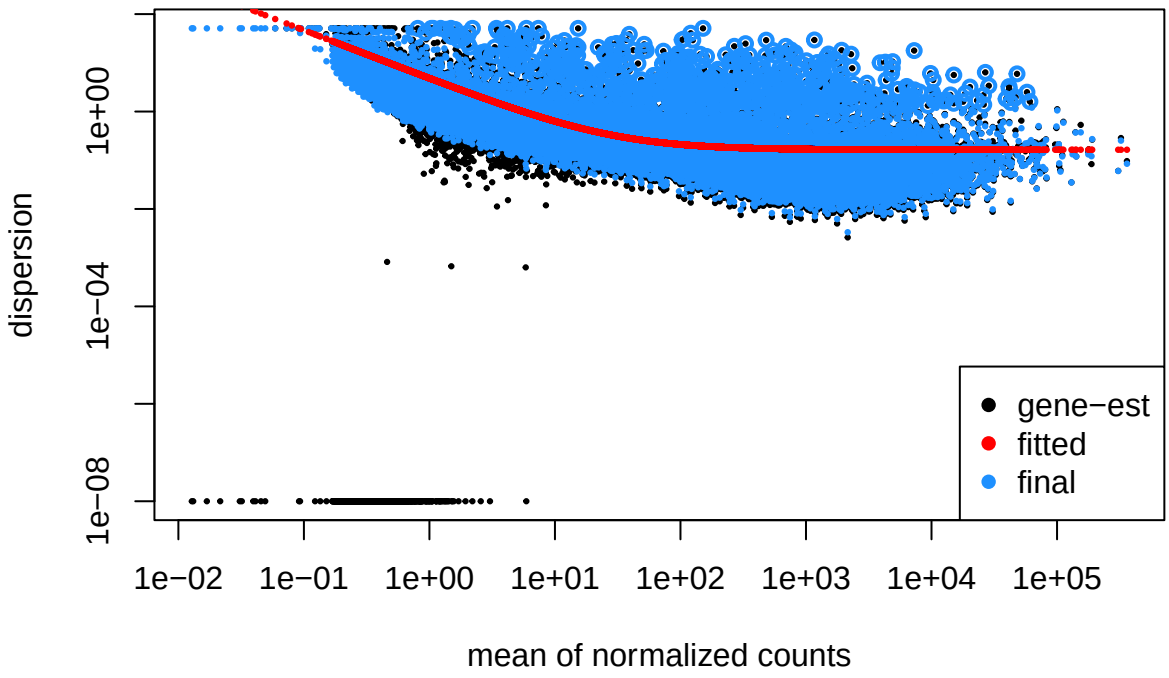
PCA plot of the samples

```
vsdata <- vst(ddsDE, blind = FALSE)
plotPCA(vsdata, intgroup= "Condition")+
  geom_label(aes(label = name)) +theme_bw()
```



Dispersion plot

```
plotDispEsts(ddsDE)
```

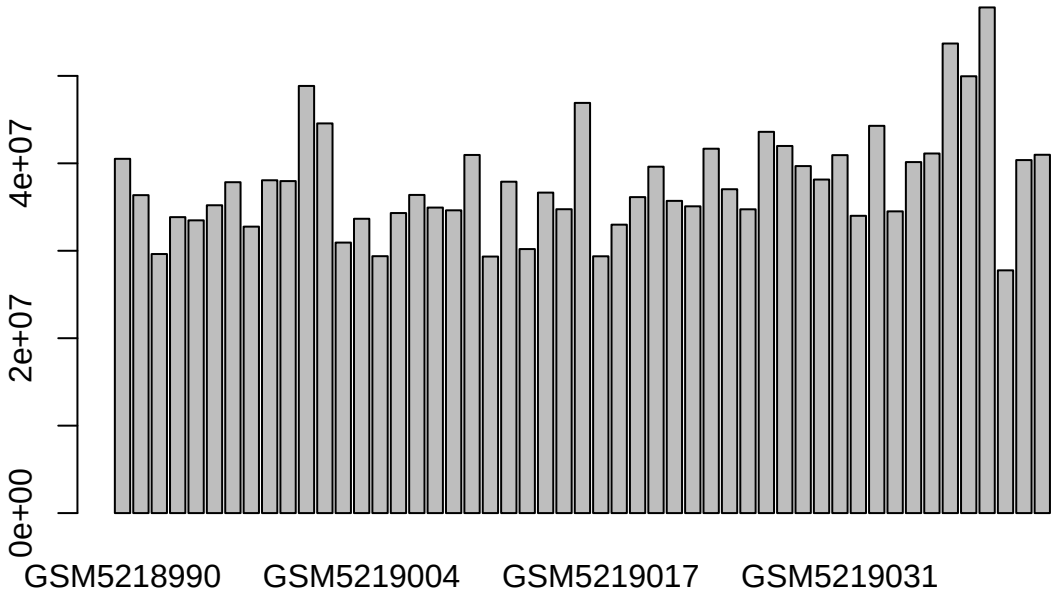


Read counts distribution across samples

```
colSums(counts(dds))
```

```
## GSM5218990 GSM5218991 GSM5218992 GSM5218993 GSM5218994 GSM5218995 GSM5218996
##   40512084   36359049   29626322   33842591   33477491   35203433   37838145
## GSM5218997 GSM5218998 GSM5218999 GSM5219001 GSM5219002 GSM5219003 GSM5219004
##   32757743   38060897   37966561   48848537   44564073   30933692   33657044
## GSM5219005 GSM5219006 GSM5219007 GSM5219008 GSM5219009 GSM5219010 GSM5219011
##   29379190   34313935   36385386   34940167   34624579   40954436   29337686
## GSM5219012 GSM5219013 GSM5219014 GSM5219015 GSM5219016 GSM5219017 GSM5219018
##   37898167   30192201   36648957   34750937   46908542   29368191   32985131
## GSM5219019 GSM5219020 GSM5219021 GSM5219022 GSM5219023 GSM5219025 GSM5219026
##   36135350   39615063   35704917   35083240   41670366   37038129   34740113
## GSM5219027 GSM5219028 GSM5219029 GSM5219030 GSM5219031 GSM5219032 GSM5219033
##   43598952   41983513   39683974   38144098   40934173   33995871   44285795
## GSM5219034 GSM5219035 GSM5219036 GSM5219037 GSM5219038 GSM5219039 GSM5219041
##   34506458   40141709   41124133   53701018   49954680   57826941   27759251
## GSM5219042 GSM5219043
##   40369542   40977741
```

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

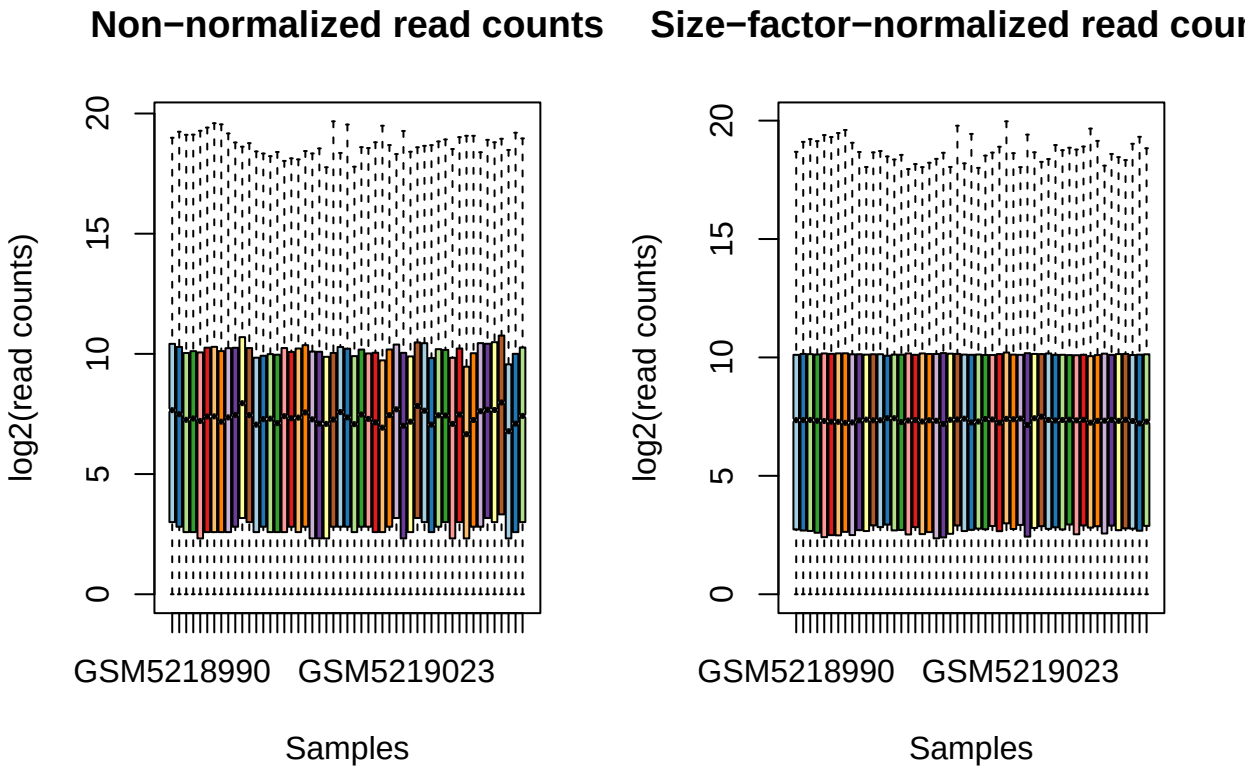



```
par(mfrow=c(1,2))
```

Boxplots of read counts before and after normalization

```
col<- brewer.pal(12, 'Paired')
par(mfrow=c(1,2))

#Before normalization
boxplot(log2(counts(ddsDE)+1), notch=TRUE,col=col,
        main = "Non-normalized read counts",  xlab="Samples",
        ylab="log2(read counts)", cex = .2)
#After normalization
boxplot(log2(counts(ddsDE, normalize= TRUE) +1), notch=TRUE, col=col,
        main = "Size-factor-normalized read counts", xlab="Samples",
           ylab="log2(read counts)", cex = .2)
```




```
graphics.off()

#Plot a single probe
plotCounts(ddsDE, gene="ENSG00000000003", intgroup="Condition")
```

Save normalized counts

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

DESeq2 Results

```
res<- results(ddsDE, alpha = 0.01)
head(res)
```

```
## log2 fold change (MLE): Condition Covid vs Control
## Wald test p-value: Condition Covid vs Control
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003    16.9782      -1.138039 0.3377403  -3.36957 7.52861e-04
```

```
## ENSG00000000419  1144.0467      -0.174486 0.0583126  -2.99225 2.76929e-03
## ENSG00000000457   927.9939      -0.134552 0.0744103  -1.80824 7.05690e-02
## ENSG00000000460   252.8247       0.489112 0.0927944   5.27092 1.35739e-07
## ENSG00000000938 34385.3376       1.006987 0.1899306   5.30186 1.14626e-07
## ENSG00000000971   177.2686      -1.154292 0.3015474  -3.82789 1.29244e-04
##                       padj
##                  <numeric>
## ENSG00000000003 2.66081e-03
## ENSG00000000419 8.18507e-03
## ENSG00000000457 1.27605e-01
## ENSG00000000460 1.47816e-06
## ENSG00000000938 1.28135e-06
## ENSG00000000971 5.76990e-04
```

```
summary(res)
```

```
##
## out of 21795 with nonzero total read count
## adjusted p-value < 0.01
## LFC > 0 (up)       : 4020, 18%
## LFC < 0 (down)     : 3161, 15%
## outliers [1]       : 0, 0%
## low counts [2]     : 1280, 5.9%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
head(res)
```

```
## log2 fold change (MLE): Condition Covid vs Control
## Wald test p-value: Condition Covid vs Control
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003    16.9782      -1.138039 0.3377403  -3.36957 7.52861e-04
## ENSG00000000419  1144.0467      -0.174486 0.0583126  -2.99225 2.76929e-03
## ENSG00000000457   927.9939      -0.134552 0.0744103  -1.80824 7.05690e-02
## ENSG00000000460   252.8247       0.489112 0.0927944   5.27092 1.35739e-07
## ENSG00000000938 34385.3376       1.006987 0.1899306   5.30186 1.14626e-07
## ENSG00000000971   177.2686      -1.154292 0.3015474  -3.82789 1.29244e-04
##                       padj
##                  <numeric>
## ENSG00000000003 2.66081e-03
## ENSG00000000419 8.18507e-03
## ENSG00000000457 1.27605e-01
## ENSG00000000460 1.47816e-06
## ENSG00000000938 1.28135e-06
## ENSG00000000971 5.76990e-04
```

```
write.csv(res, file = 'DEGs.csv')
```

Make contrast comparison

resultsNames(ddsDE) #In case there is multiple group generate results specific coeficient #res.contrast <- results(dds, name="condition_A_vs_ctl") #Name = condition name

Order DEGs

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

Annotation

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

```
## '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>
## ENSG00000143603   312.802        4.49015  0.361428   12.4234 1.95199e-35
## ENSG00000164587 10397.066       -1.61180  0.130077  -12.3911 2.91982e-35
## ENSG00000156482 19422.285       -1.24164  0.102326  -12.1341 6.96657e-34
## ENSG00000164045   240.400        3.27877  0.270047   12.1415 6.36654e-34
## ENSG00000075218   252.717        2.86287  0.239843   11.9364 7.64276e-33
## ENSG00000101057  2002.515        3.51944  0.295907   11.8937 1.27614e-32
##                         padj      symbol
##                    <numeric> <character>
## ENSG00000143603 2.99676e-31       KCNN3
## ENSG00000164587 2.99676e-31       RPS14
## ENSG00000156482 3.57507e-30       RPL30
## ENSG00000164045 3.57507e-30      CDC25A
## ENSG00000075218 3.13766e-29       GTSE1
## ENSG00000101057 4.36588e-29       MYBL2
```

Stratify

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

Remove duplicates and Save

```
#na.omit(res_final)
Significant_DEGs<- res_final[!duplicated(res_final$symbol),]
write.csv(Significant_DEGs, file = 'Significant_DEGs.csv')
```

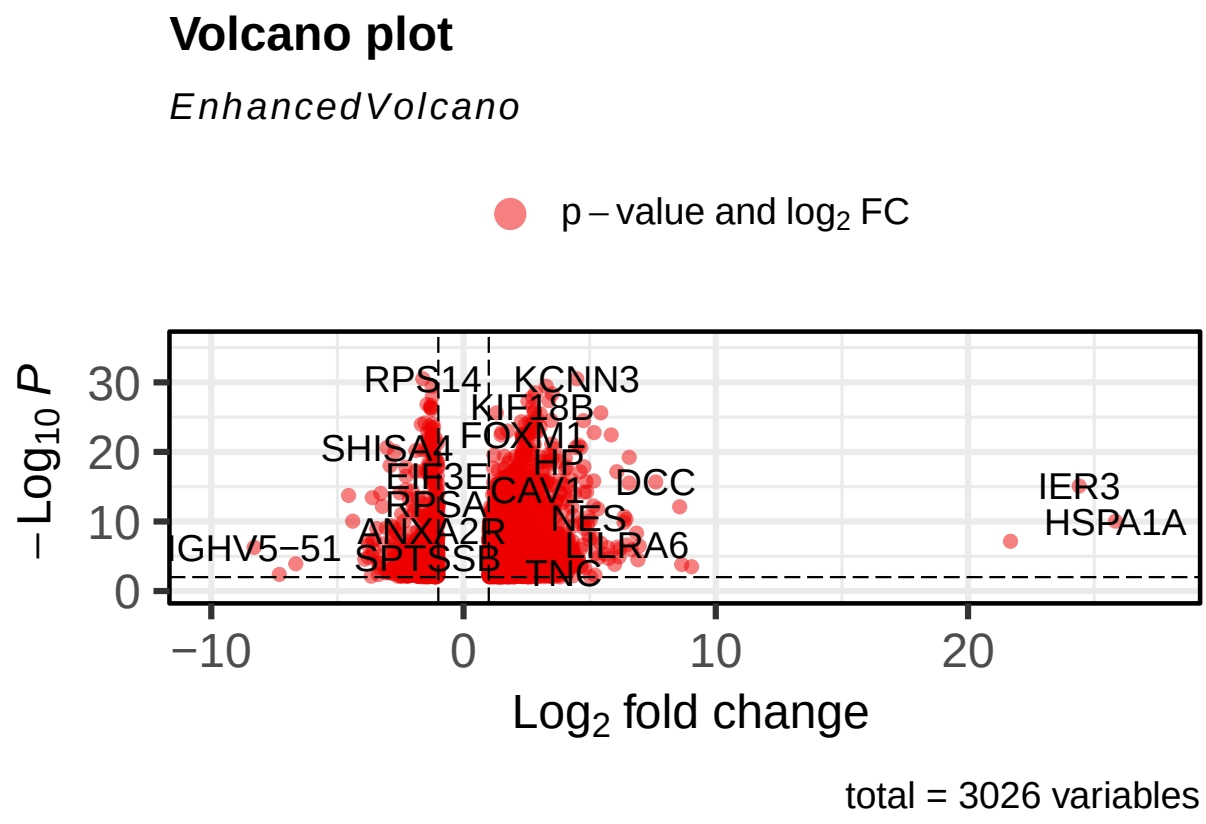
Save Significant UPs and Down

```
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')
```

Volcano Plot

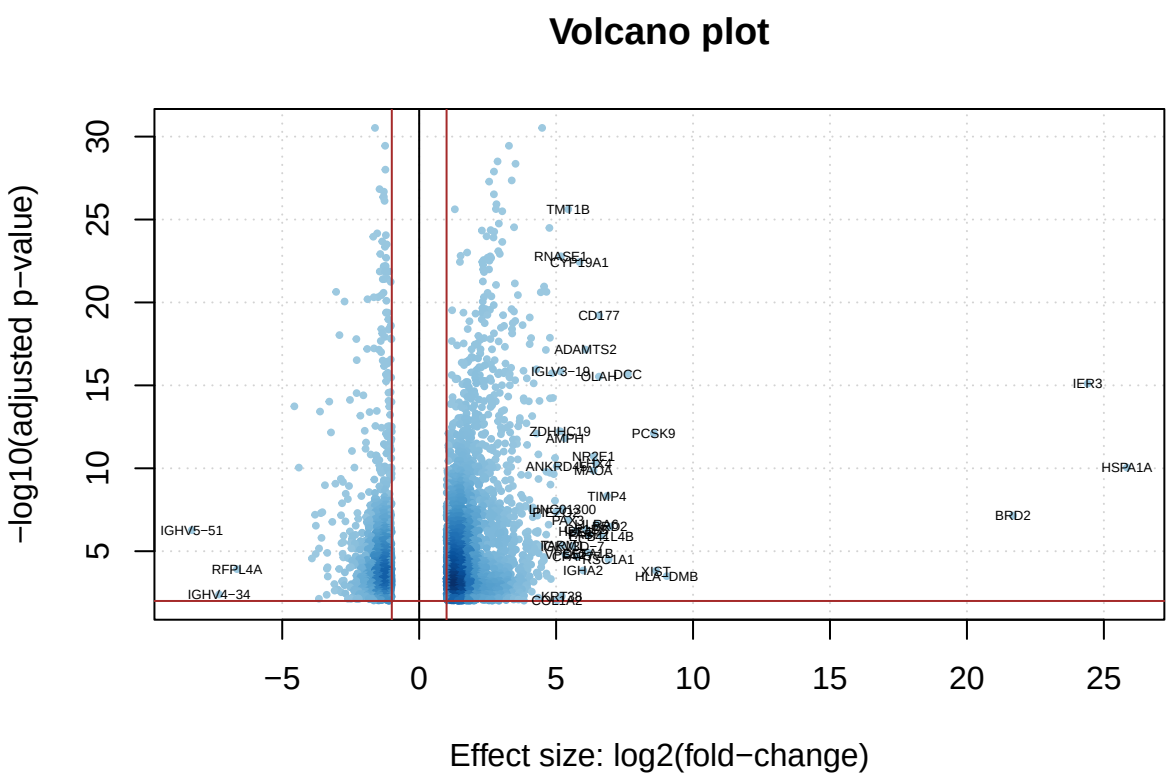
```
#Method 1
Plot.df<-data.frame(res_final)

EnhancedVolcano(Plot.df, x="log2FoldChange", y="padj", lab = Plot.df$symbol,
                pCutoff = 0.01, FCcutoff = 1, border = 'full')
```



```
#Method 2
#jpeg('Volcano_plot.jpeg',  width = 6, height = 4, units = 'in', res  = 1200)
alpha <- 0.01 # Threshold on the adjusted p-value
cols <- densCols(res_final$log2FoldChange, -log10(res_final$pvalue))
plot(res_final$log2FoldChange, -log10(res_final$padj), col=cols, panel.first=grid(),
     main="Volcano plot", xlab="Effect size: log2(fold-change)", ylab="-log10(adjusted p-value)",
     pch=20, cex=0.6)
abline(v=0)
abline(v=c(-1,1), col="brown")
abline(h=-log10(alpha), col="brown")

gn.selected <- abs(res_final$log2FoldChange) > 5 & res_final$padj < alpha
text(res_final$log2FoldChange[gn.selected],
     -log10(res_final$padj)[gn.selected],
     lab=res_final$symbol[gn.selected ], cex=0.4)
```




```
#dev.off()
```