RNAseq turtorial for DEG analysis

Amarinder Singh Thind

08 Oct, 2020

Contents

```
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)
```

Install and load packages

```
# if (!requireNamespace('BiocManager', quietly = TRUE))
# install.packages('BiocManager')

# BiocManager::install('DESeq2')
# BiocManager::install('edgeR')
# BiocManager::install('biomaRt')
# BiocManager::install('PCAtools')

library(edgeR)
library(DESeq2)
library("biomaRt")
```

load the raw count matrix

```
setwd("./")

rawcount <- read.table("RawGeneCounts.tsv", header = TRUE, sep = "\t",
    row.names = 1)
```

Filter for coding genes ## or skip

```
mart <- useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
all_coding_genes <- getBM(attributes = c("hgnc_symbol"), filters = c("biotype"),
    values = list(biotype = "protein_coding"), mart = mart)
rawcount <- rawcount[row.names(rawcount) %in% all_coding_genes$hgnc_symbol,
    ]
```

Meta data/ Data annotation

```
anno <- read.table("Annotation_of_samples.csv", header = TRUE,
    sep = ",")  ##In this case Two coulmns (a) sample (b) Condition
rownames(anno) <- anno$sample
```

Define conditions (for contrast) that you want to compare if you have more than one #control #case

This is pair-wise comparison, so only consider one pair at one time

```
firstC <- "case1"  #case1 #case2 #case3 etc
SecondC <- "Control"
p.threshold <- 0.05  ##define threshold for filtering
```

subset raw and conditional data for defined pairs

```
anno <- anno[(anno$Condition == firstC | anno$Condition == SecondC),
    ]
rawcount <- rawcount[, names(rawcount) %in% anno$sample]
```

DESeq2

create Desq2 datasets

```
dds <- DESeqDataSetFromMatrix(countData = rawcount, colData = anno,
    design = ~Condition)
```

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

```
##   it appears that the last variable in the design formula, 'Condition',
##   has a factor level, 'Control', which is not the reference level. we recommend
##   to use factor(...,levels=...) or relevel() to set this as the reference level
##   before proceeding. for more information, please see the 'Note on factor levels'
##   in vignette('DESeq2').
```

Run DESEQ2

```
dds <- 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 32 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

contrast based comparison

```
# In case of multiple comparisons ## we need to change the
# contrast for every comparision

contrast <- c("Condition", firstC, SecondC)
res <- results(dds, contrast = contrast)
res$threshold <- as.logical(res$padj < p.threshold)  #Threshold defined earlier

nam <- paste("down_in", firstC, sep = "_")
# res$nam <- as.logical(res$log2FoldChange < 0)
res[, nam] <- as.logical(res$log2FoldChange < 0)

genes.deseq <- row.names(res)[which(res$threshold)]

genes_deseq2_sig <- res[which(res$threshold), ]

file <- paste("Deseq2_", firstC, "_v_", SecondC, "_results_significant_padj0.05.csv",
    sep = "")
all_results <- paste("Deseq2_", firstC, "_v_", SecondC, "_all_results.csv",
    sep = "")

write.table(genes_deseq2_sig, file, sep = ",")
write.table(res, all_results, sep = ",")
```

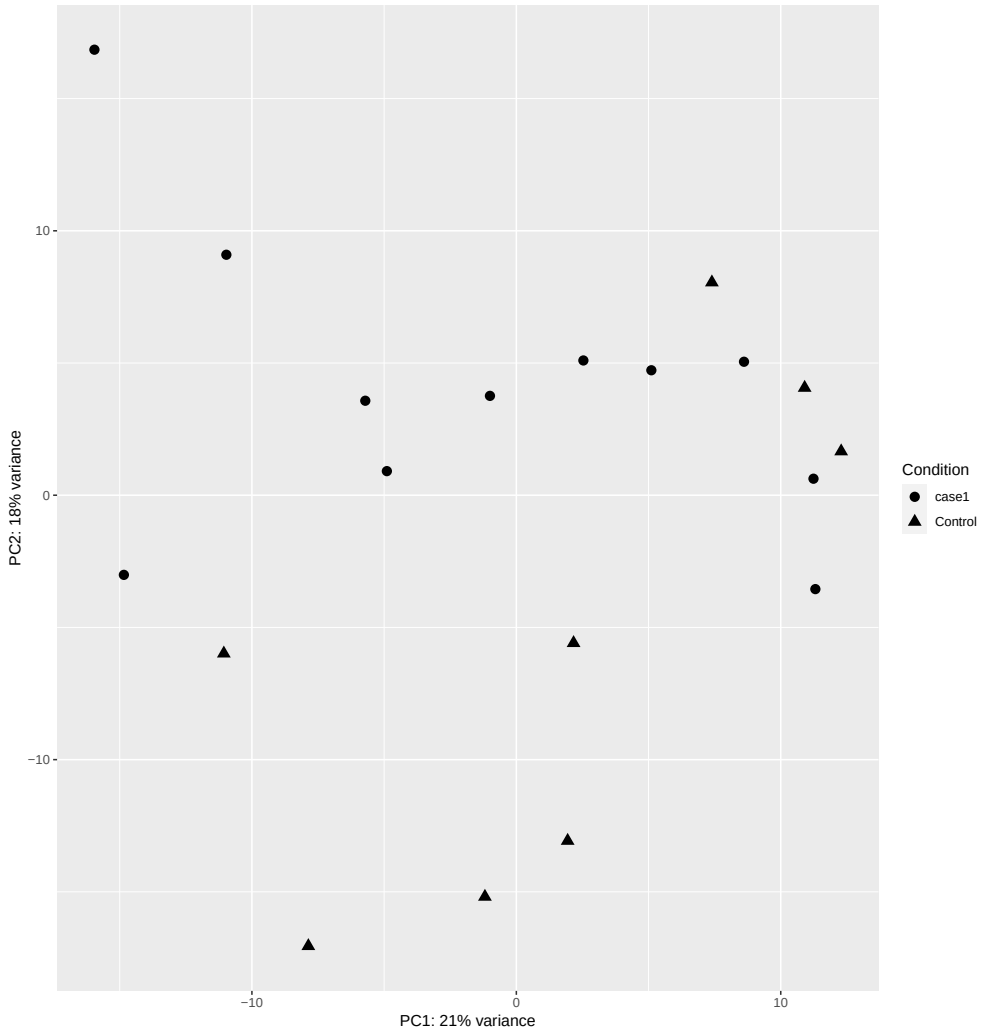
PCA and Heat-MAp Plots

Varinace transformation vst or rlog

```
vsd <- vst(dds, blind = FALSE)  #Variance type (a) Vst or (b) rlog
# rld <- rlog(dds, blind=FALSE)
```

PCA (Deseq2) with design consideration (Consider top 500 highest variable genes)

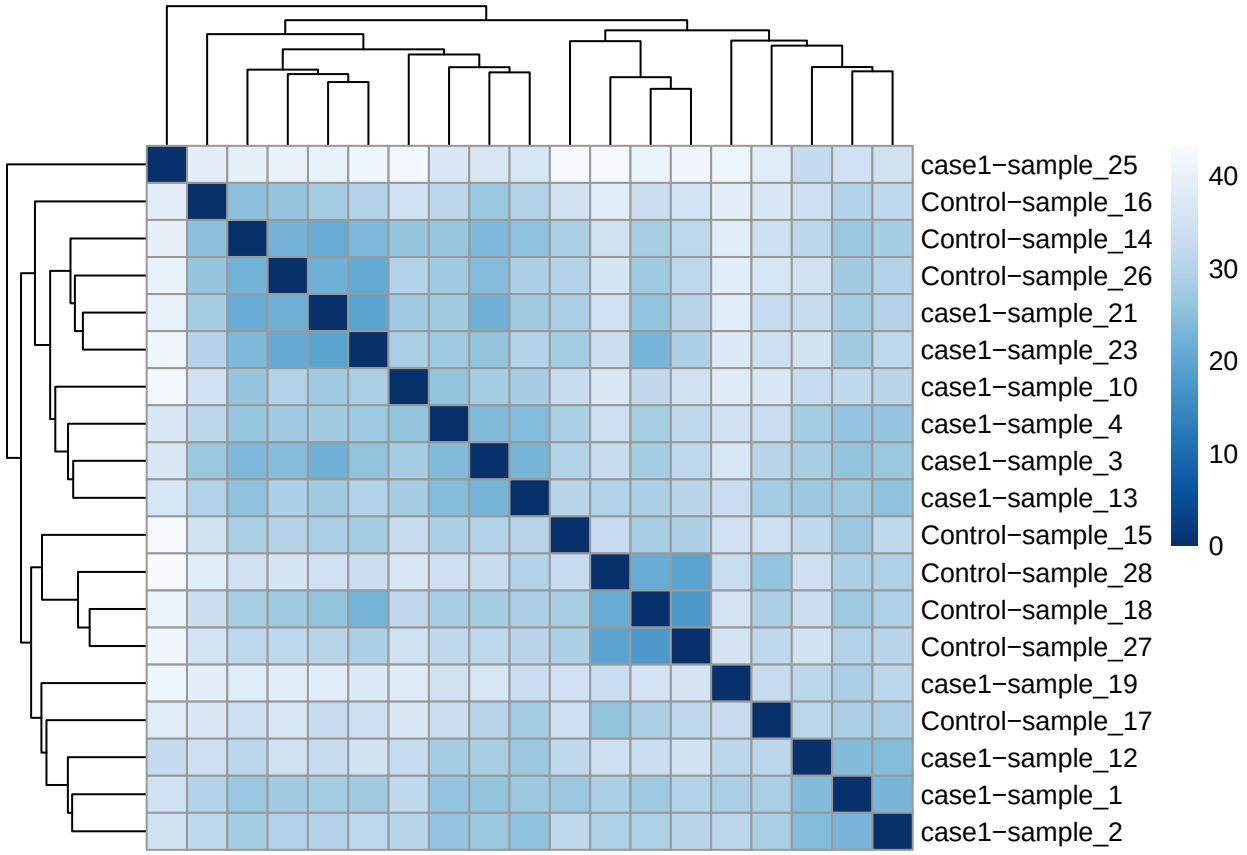
```
library(ggplot2)
pcaData <- plotPCA(vsd, intgroup=c("Condition", "sample"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2,  shape=Condition)) + #color=sample,
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed()
```



heatmap

```
sampleDists <- dist(t(assay(vsd)))
library("RColorBrewer")
library("pheatmap")
sampleDistMatrix <- as.matrix(sampleDists)

rownames(sampleDistMatrix) <- paste(vsd$Condition, vsd$sample,
```

```
    sep = "-")

colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette(rev(brewer.pal(9, "Blues")))(255)
pheatmap(sampleDistMatrix, clustering_distance_rows = sampleDists,
    clustering_distance_cols = sampleDists, col = colors)
```



edgeR

build edgeR data

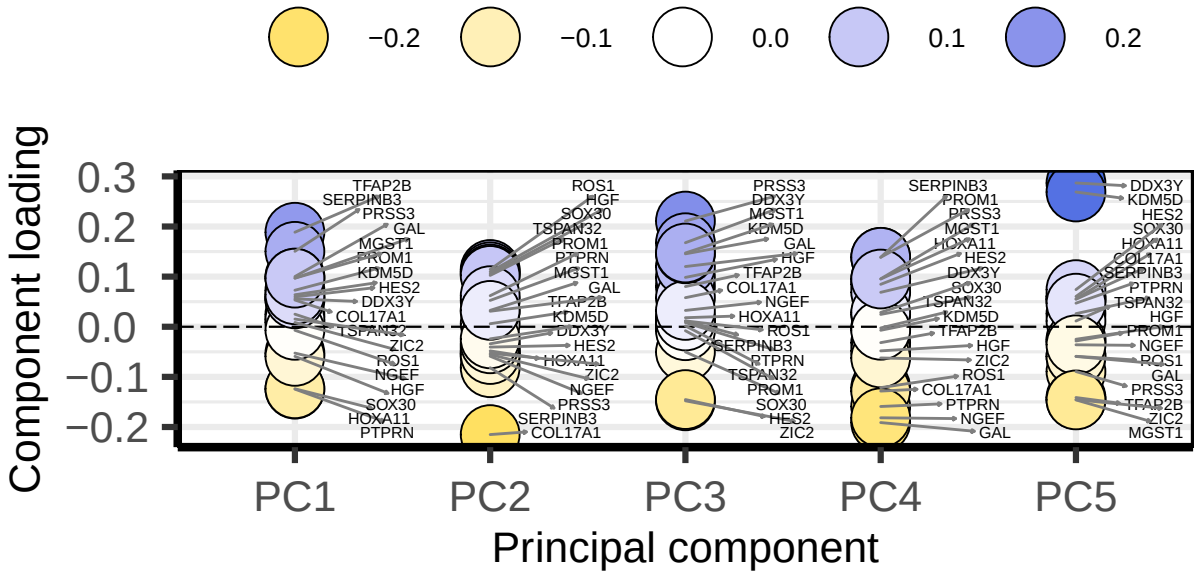
```
dge <- DGEList(counts = rawcount, group = anno$Condition)
```

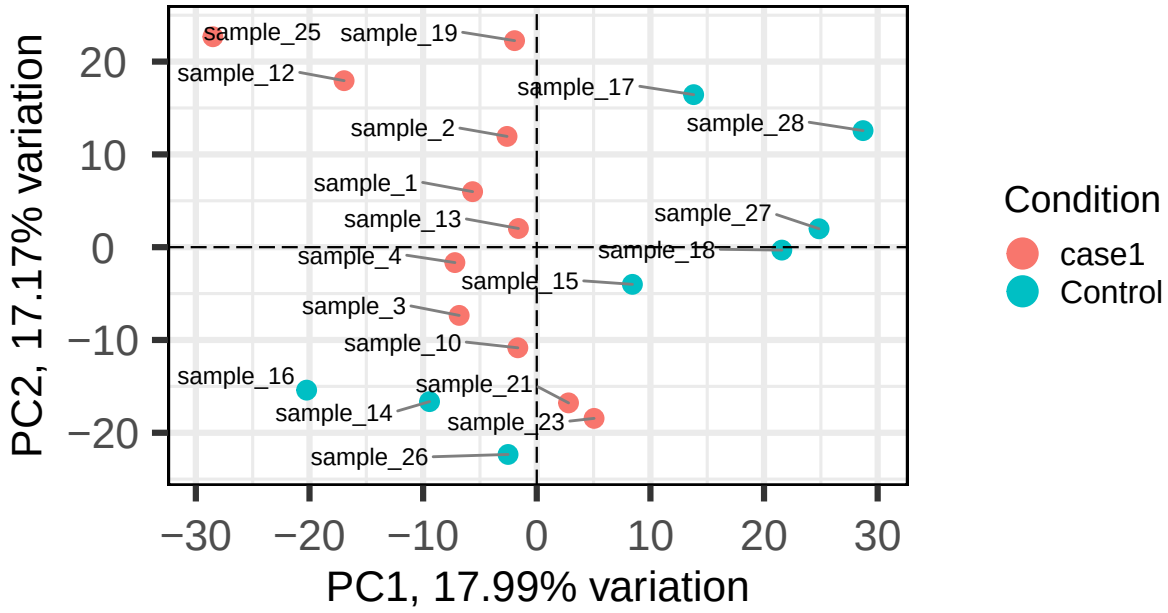
Normalization and PCA plot

PCA ## for more details, please visit following link https://bioconductor.org/packages/release/bioc/vignettes/PCAtools/inst/doc/PCAtools.html

```
## Loading required package: ggrepel
```

```
##
## Attaching package: 'PCAtools'

## The following objects are masked from 'package:stats':
##
##     biplot, screeplot

## -- removing the lower 20% of variables based on variance

## -- variables retained:

## TFAP2B, PTPRN, HOXA11, ROS1, HGF, SOX30, TSPAN32, COL17A1, PRSS3, ZIC2, HES2, PROM1, SERPINB3, GAL,
```





filter out lowly expressed genes

```
keep <- filterByExpr(dge)
dge <- dge[keep, , keep.lib.sizes = FALSE]
# It is recommended to recalculate the library sizes of the
# DGEList object after the filtering, although the downstream
# analysis is robust to whether this is done or not.
```

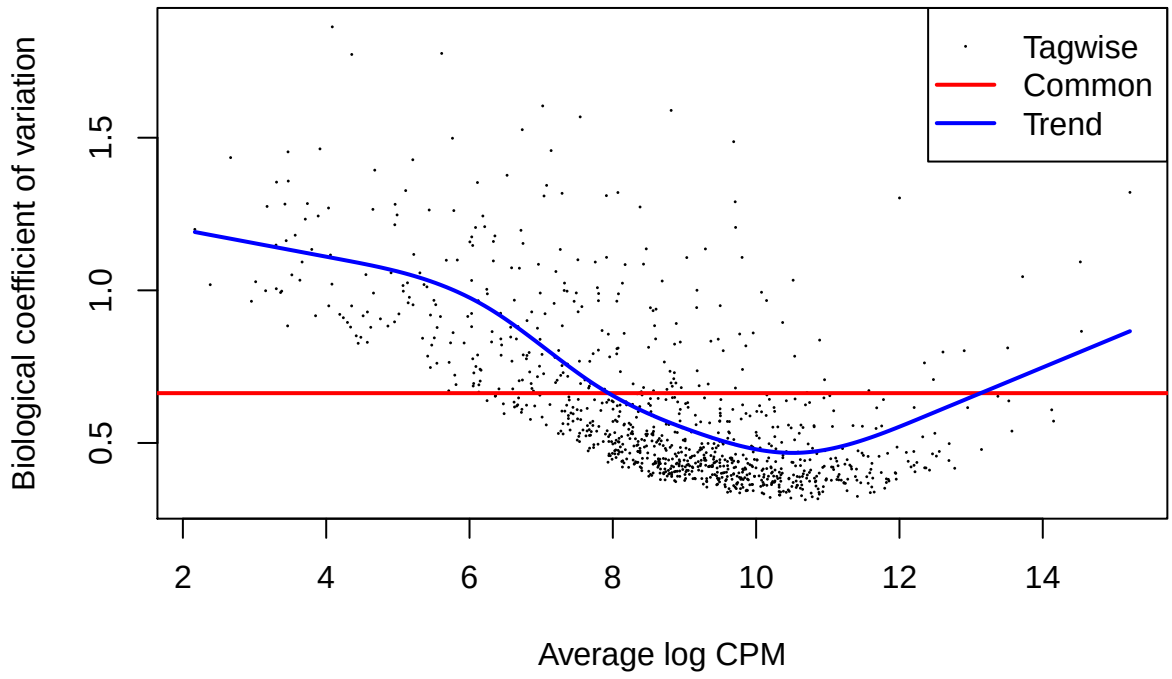
Create the contrast matrix

```
##    case1 Control
## 1      1       0
## 2      1       0
## 3      1       0
## 4      1       0
## 5      1       0
## 6      1       0
## 7      1       0
## 8      0       1
## 9      0       1
## 10     0       1
## 11     0       1
## 12     0       1
## 13     1       0
## 14     1       0
```

```
## 15     1       0
## 16     1       0
## 17     0       1
## 18     0       1
## 19     0       1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$`dge$samples$group`
## [1] "contr.treatment"
```

Estimate dispersion parameter for GLM

```
dge <- estimateGLMCommonDisp(dge, design.mat)
dge <- estimateGLMTrendedDisp(dge, design.mat)
dge <- estimateGLMTagwiseDisp(dge, design.mat)
# Plot mean-variance
plotBCV(dge)
```



Model fitting

```
fit.edgeR <- glmQLFit(dge, design.mat)

# Differential expression
```

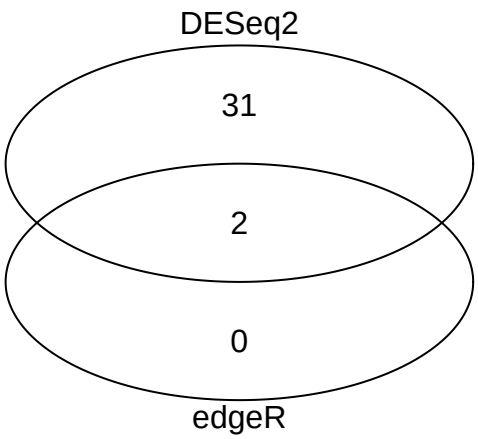
```
contrasts.edgeR <- makeContrasts(case1 - Control, levels = design.mat)  ##FirstC-SecondC ##Define


qlf.edgeR <- glmQLFTest(fit.edgeR, contrast = contrasts.edgeR)

##### DGE at padjust 0.05

# Access results tables
edgeR_results <- qlf.edgeR$table
sig.edgeR <- decideTestsDGE(qlf.edgeR, adjust.method = "BH",
    p.value = p.threshold)
# View(sig.edgeR)
significant_table <- edgeR_results[which(sig.edgeR != 0), ]
significant_table$gene <- row.names(significant_table)
genes.edgeR <- row.names(edgeR_results)[which(sig.edgeR != 0)]

edgeR_results$genes <- row.names(edgeR_results)


file_sigTab <- paste("edgeR_", firstC, "_v_", SecondC, "_results_significant_padj0.05.csv",
    sep = "")
file_allRes <- paste("edgeR_", firstC, "_v_", SecondC, "_all_results.csv",
    sep = "")

write.table(significant_table, file_sigTab, sep = ",")
write.table(edgeR_results, file_allRes, sep = ",")
```

Overlapped genes between deseq2 and edgeR

```
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
```

```
venn(list(edgeR = genes.edgeR, DESeq2 = genes.deseq))
```




```
overlapped_genes <- intersect(genes.deseq, genes.edgeR)


file_common <- paste("Common_DEG_deseq2_edgeR_", firstC, "_v_",
    SecondC, ".csv", sep = "")
write.table(overlapped_genes, file_common, sep = ",", row.names = F)
```

Quick enrichment analysis

```
# BiocManager::install('ReactomePA')
library(ReactomePA)
```

```
##
```

```
## Registered S3 method overwritten by 'enrichplot':
##   method               from
##   fortify.enrichResult DOSE
```

```
## ReactomePA v1.30.0  For help: https://guangchuangyu.github.io/ReactomePA
##
## If you use ReactomePA in published research, please cite:
## Guangchuang Yu, Qing-Yu He. ReactomePA: an R/Bioconductor package for reactome pathway analysis and
```

```
all <- overlapped_genes  ## retreive EntrezGene id's

genes = getBM(attributes = c("hgnc_symbol", "entrezgene_id"),
    filters = "hgnc_symbol", values = all, bmHeader = T, mart = mart)
```

```
## Cache found
```

```
genes1 <- genes$`NCBI gene (formerly Entrezgene) ID`



# ?enrichPathway #pvalueCutoff=0.02, #pAdjustMethod = 'BH',
# qvalueCutoff = 0.01,
x <- enrichPathway(gene = genes1, pvalueCutoff = 0.05, readable = T)
```

```
## Loading required package: org.Hs.eg.db
```

```
## Loading required package: AnnotationDbi
```

```
##
```

```
## --> No gene can be mapped....
```

```
## --> Expected input gene ID: 10162,768,5724,3293,7441,353376
```

```
## --> return NULL...
```

```
# head(as.data.frame(x)) barplot(x, showCategory=10)
# dotplot(x, showCategory=10) emapplot(x) cnetplot(x,
# categorySize='pvalue', foldChange=genes1) emapplot(x,
# color='pvalue') viewPathway('Extracellular matrix
# organization', readable=TRUE, foldChange=genes1) ## it's an
# example
```