Part 2

## 1. Generate experiment design table and store in csv file

library(DESeq2)

## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##   
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':  
##   
## clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
## clusterExport, clusterMap, parApply, parCapply, parLapply,  
## parLapplyLB, parRapply, parSapply, parSapplyLB

## 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, cbind, colMeans,  
## colnames, colSums, do.call, duplicated, eval, evalq, Filter,  
## Find, get, grep, grepl, intersect, is.unsorted, lapply,  
## lengths, Map, mapply, match, mget, order, paste, pmax,  
## pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce,  
## rowMeans, rownames, rowSums, sapply, setdiff, sort, table,  
## tapply, union, unique, unsplit, which, which.max, which.min

##   
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':  
##   
## expand.grid

## Loading required package: IRanges

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## 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")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##   
## Attaching package: 'matrixStats'

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

##   
## Attaching package: 'DelayedArray'

## The following objects are masked from 'package:matrixStats':  
##   
## colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges

## The following object is masked from 'package:base':  
##   
## apply

# Load in count matrix   
countTable <- read.csv("C:\\Users\\2060576e\\Downloads\\RNA-seq assessment\\gene\_count\_matrix.csv", row.names=1)  
  
  
group\_names <-c(rep("A", 4), rep("B", 4), rep("C", 4 ))  
  
# Make experiment design table  
design\_col <- c("Samples", "Group")  
sample\_col <- colnames(countTable)  
design <- data.frame(Sample=sample\_col, Group=group\_names)  
   
write.csv(design, file="Design-Exp.csv", row.names = FALSE, sep="\t", quote = FALSE)

## Warning in write.csv(design, file = "Design-Exp.csv", row.names = FALSE, :  
## attempt to set 'sep' ignored

## 2. Generate DEseqDataSet (dds) object

# Assign groups  
Group <- factor(c(rep("A", 4), rep("B", 4), rep("C", 4 )))  
  
# Make data frame for DESeqDataSet colData  
coldata <- read.csv("Design-Exp.csv", row.names = 1)  
  
# Create DESegDataSet object  
dds <- DESeqDataSetFromMatrix(countData=countTable, colData=coldata, design=~Group)  
dds

## class: DESeqDataSet   
## dim: 4382 12   
## metadata(1): version  
## assays(1): counts  
## rownames(4382): Gpr158 MSTRG.3594 ... MSTRG.3072 MSTRG.1173  
## rowData names(0):  
## colnames(12): A1 A2 ... C3 C4  
## colData names(1): Group

resultsNames(dds)

## character(0)

## 3. Run differential Expression Analysis

# Fillter genes that have zero-count  
notAllZero <- (rowSums(counts(dds)) >0)  
dds <- dds[notAllZero]  
dds

## class: DESeqDataSet   
## dim: 3749 12   
## metadata(1): version  
## assays(1): counts  
## rownames(3749): MSTRG.3594 Gm14023 ... MSTRG.3072 MSTRG.1173  
## rowData names(0):  
## colnames(12): A1 A2 ... C3 C4  
## colData names(1): Group

# Run DE analysis  
dds <- DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

Inspecting Heteroskedacity in count data with dispersion plot, and plotting PCA of rlog transformed counts

# Plotting mean-variance relationship (dispersion plot)  
dds <- estimateSizeFactors(dds)  
dds <- estimateDispersions(dds)

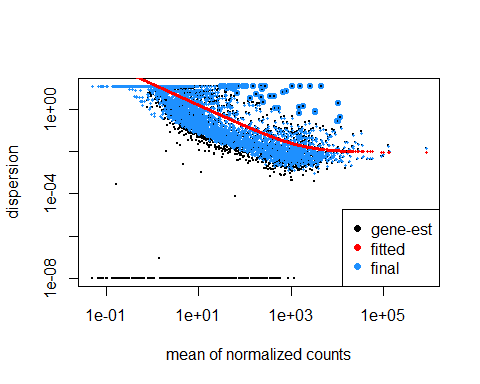
## found already estimated dispersions, replacing these

## gene-wise dispersion estimates

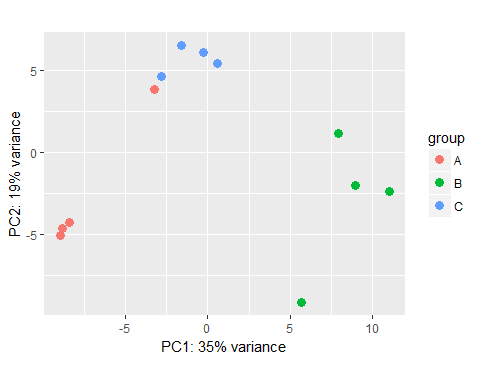
## mean-dispersion relationship

## final dispersion estimates

plotDispEsts(dds)



# Using regularized-logarithm transformstion (rolg) to stabilize the variance across the mean  
rld <- rlog(dds)  
  
  
# PCA plot of rlog transformed data  
plotPCA(rld, intgroup="Group")



## 4. Continuing DE analysis form the DESeqDataSet

dds <- DESeq(dds)

## 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

resdata <- as.data.frame(counts(dds, normalized=TRUE))  
resdata <- data.frame(IDs = row.names(resdata), resdata)  
  
# DE analysis of group A vs B  
res <- results(dds, contrast = c("Group", "B", "A"))  
  
### Order by adjusted p-value  
res <- res[order(res$padj), ]   
summary(res)

##   
## out of 3749 with nonzero total read count  
## adjusted p-value < 0.1  
## LFC > 0 (up) : 415, 11%   
## LFC < 0 (down) : 368, 9.8%   
## outliers [1] : 76, 2%   
## low counts [2] : 1328, 35%   
## (mean count < 8)  
## [1] see 'cooksCutoff' argument of ?results  
## [2] see 'independentFiltering' argument of ?results

### Convert to a data-frame, with a column at the start called "IDs" for the gene IDs  
resdata <- data.frame(IDs = row.names(as.data.frame(res)), as.data.frame(res))  
  
  
  
# DE analysis of group B vs C  
res2 <- results(dds, contrast = c("Group", "C", "B"))  
  
### Order by adjusted p-value  
res2 <- res2[order(res2$padj), ]   
summary(res2)

##   
## out of 3749 with nonzero total read count  
## adjusted p-value < 0.1  
## LFC > 0 (up) : 338, 9%   
## LFC < 0 (down) : 416, 11%   
## outliers [1] : 76, 2%   
## low counts [2] : 1186, 32%   
## (mean count < 7)  
## [1] see 'cooksCutoff' argument of ?results  
## [2] see 'independentFiltering' argument of ?results

### Convert to a data-frame, with a column at the start called "IDs" for the gene IDs  
resdata2 <- data.frame(IDs = row.names(as.data.frame(res2)), as.data.frame(res2))  
  
  
  
#DE analysis of group A vs C  
res3 <- results(dds, contrast = c("Group", "C", "A"))  
  
### Order by adjusted p-value  
res3 <- res3[order(res$padj), ]   
summary(res3)

##   
## out of 3749 with nonzero total read count  
## adjusted p-value < 0.1  
## LFC > 0 (up) : 24, 0.64%   
## LFC < 0 (down) : 20, 0.53%   
## outliers [1] : 76, 2%   
## low counts [2] : 432, 12%   
## (mean count < 2)  
## [1] see 'cooksCutoff' argument of ?results  
## [2] see 'independentFiltering' argument of ?results

### Convert to a data-frame, with a column at the start called "IDs" for the gene IDs  
resdata3 <- data.frame(IDs = row.names(as.data.frame(res3)), as.data.frame(res3))

## 5. Write DE results to csv file

library(readr)  
  
# A vs B  
write\_csv(resdata, "C:\\Users\\2060576e\\Documents\\A\_vs\_B.csv", append=FALSE)  
  
# B vs C  
write\_csv(resdata2, "C:\\Users\\2060576e\\Documents\\B\_vs\_C.csv", append=FALSE)  
  
# A vs C  
write\_csv(resdata3, "C:\\Users\\2060576e\\Documents\\A\_vs\_C.csv", append=FALSE)