RNA Analysis 23

Michael Callahan

2024-04-03

Load data

```r
library(here)
```

```
## Warning: package 'here' was built under R version 4.3.3
```

```
## here() starts at C:/Users/mgcal/OneDrive/Documents/School/Courses/Stat 555/Project/Stat555Project
```

```r
library(dplyr)
```

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

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

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

```r
current_project_path <- here()

file_path <- file.path(current_project_path, "Data", "CMP_RNA_1.tsv")
CMP_RNA_1 <- read.delim(file_path)

file_path <- file.path(current_project_path, "Data", "CMP_RNA_2.tsv")
CMP_RNA_2 <- read.delim(file_path)

file_path <- file.path(current_project_path, "Data", "CFU-E_RNA_1.tsv")
CFUE_RNA_1 <- read.delim(file_path)

file_path <- file.path(current_project_path, "Data", "CFU-E_RNA_2.tsv")
CFUE_RNA_2 <- read.delim(file_path)
```

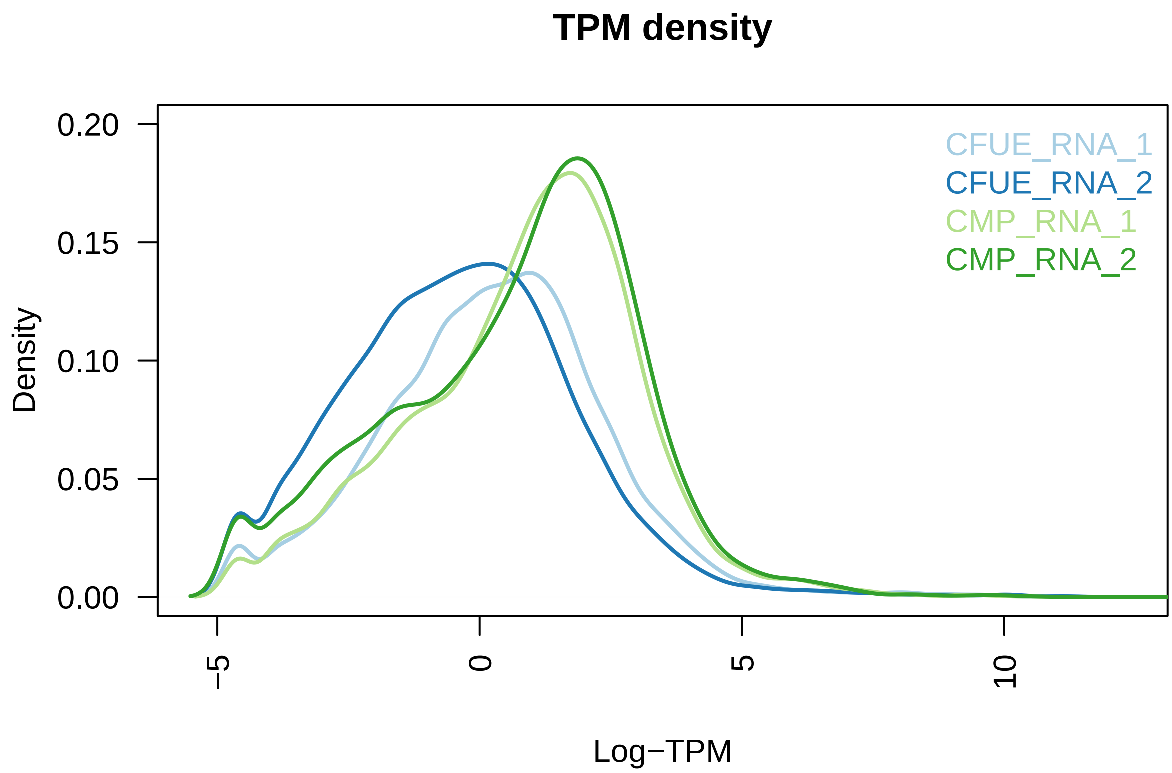
Data preprocessing

```r
#FYI - gene_ids are unique

#CREATE A TPM Dataframe
# Add prefixes to the tpm columns in each dataframe
CFUE_RNA_1_tpm <- CFUE_RNA_1 %>% select(gene_id, CFUE_RNA_1_tpm = TPM)
CFUE_RNA_2_tpm <- CFUE_RNA_2 %>% select(gene_id, CFUE_RNA_2_tpm = TPM)
CMP_RNA_1_tpm <- CMP_RNA_1 %>% select(gene_id, CMP_RNA_1_tpm = TPM)
CMP_RNA_2_tpm <- CMP_RNA_2 %>% select(gene_id, CMP_RNA_2_tpm = TPM)

# Join the dataframes together on gene_id
tpm <- inner_join(CFUE_RNA_1_tpm, CFUE_RNA_2_tpm, by = "gene_id") %>%
       inner_join(CMP_RNA_1_tpm, by = "gene_id") %>%
       inner_join(CMP_RNA_2_tpm, by = "gene_id")

rm(CFUE_RNA_1_tpm, CFUE_RNA_2_tpm, CMP_RNA_1_tpm, CMP_RNA_2_tpm)

tpm <- tpm %>%
  rename_at(vars(2:5), ~ sub("_tpm$", "", .))


# Set 'column_name' as row names
rownames(tpm) <- tpm$gene_id

# Remove 'column_name' from dataframe (optional)
tpm <- tpm[, -which(names(tpm) == 'gene_id')]

#Create a colData matrix
colData = data.frame(
  source_name = c('CMP_RNA_1','CMP_RNA_2','CFUE_RNA_1','CFUE_RNA_2'),
  group = c('CMP', 'CMP', 'CFUE', 'CFUE')
  )
```

Histogram overlay

Here, we can see a density plot of the 4 samples being compared.

```r
library(RColorBrewer)
tpm_filtered <- tpm[rowSums(tpm != 0) > 0, ]
samplenames <- colnames(tpm_filtered)
tpm.cutoff <- log2(0.1)
nsamples <- ncol(tpm_filtered)
col <- brewer.pal(nsamples, "Paired")
par(mfrow=c(1,1))
plot(density(log(tpm_filtered[,1])), col=col[1], lwd=2, ylim=c(0,0.2), las=2, main="", xlab="")
title(main="TPM density", xlab="Log-TPM")
for (i in 2:nsamples){
den <- density(log(tpm_filtered[,i]))
lines(den$x, den$y, col=col[i], lwd=2)
}
legend("topright", samplenames, text.col=col, bty="n")
```



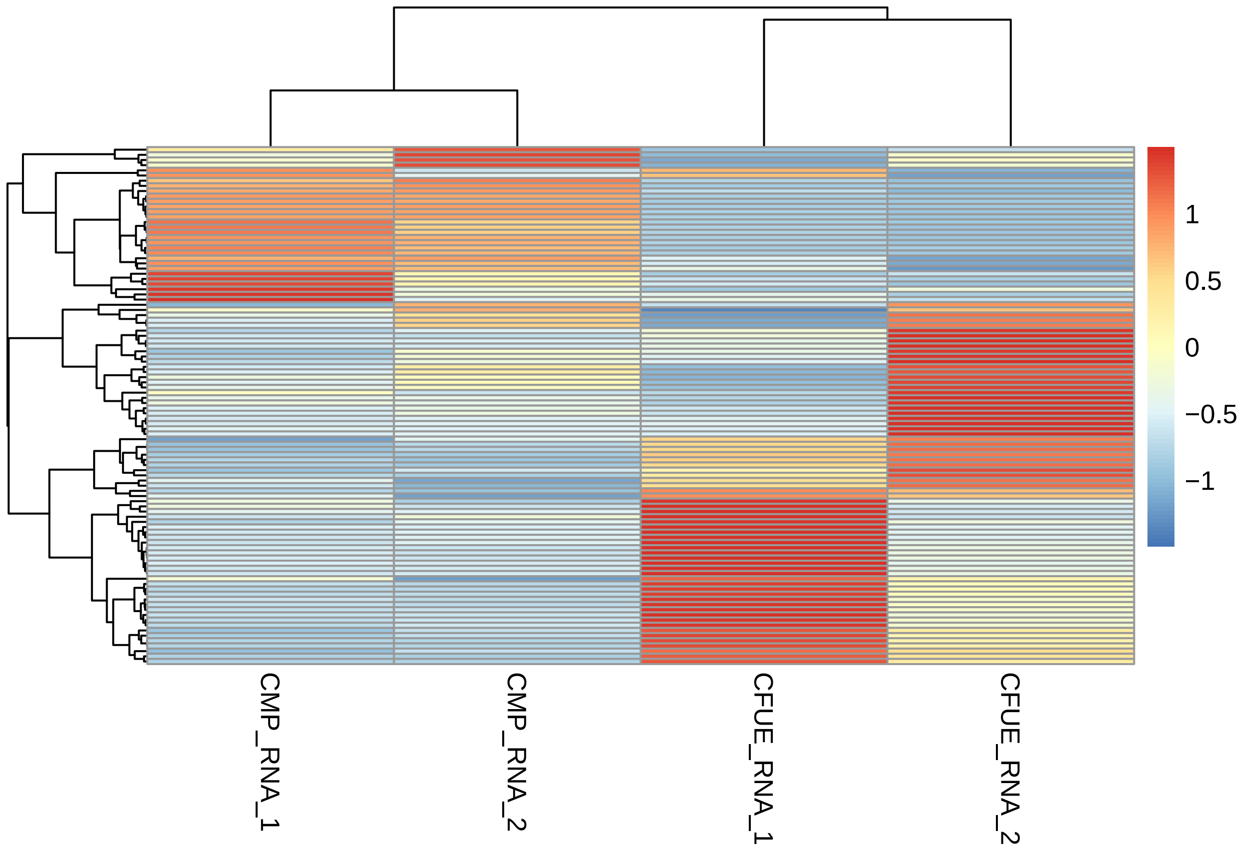
EDA - Heat map

Here, we look at the TPM (trascripts per million) for our two cell lines (2 replicates each), and we pull out the top 100 genes with the highest variance of expression across samples. Then we plot their normalized expression levels across the samples using a heat map. We should see that the replicates within each cell line should have a more similar expression pattern than across cell lines.

```
library(pheatmap)
```

```
## Warning: package 'pheatmap' was built under R version 4.3.3
```

```
#compute the variance of each gene across samples
variances <- apply(tpm, 1, var)

selectedGenes <- order(variances, decreasing = TRUE)[1:100]
pheatmap(tpm[selectedGenes,], scale = 'row', show_rownames = FALSE)
```



EDA - PCA

...

```
library(stats)
library(ggplot2)
```

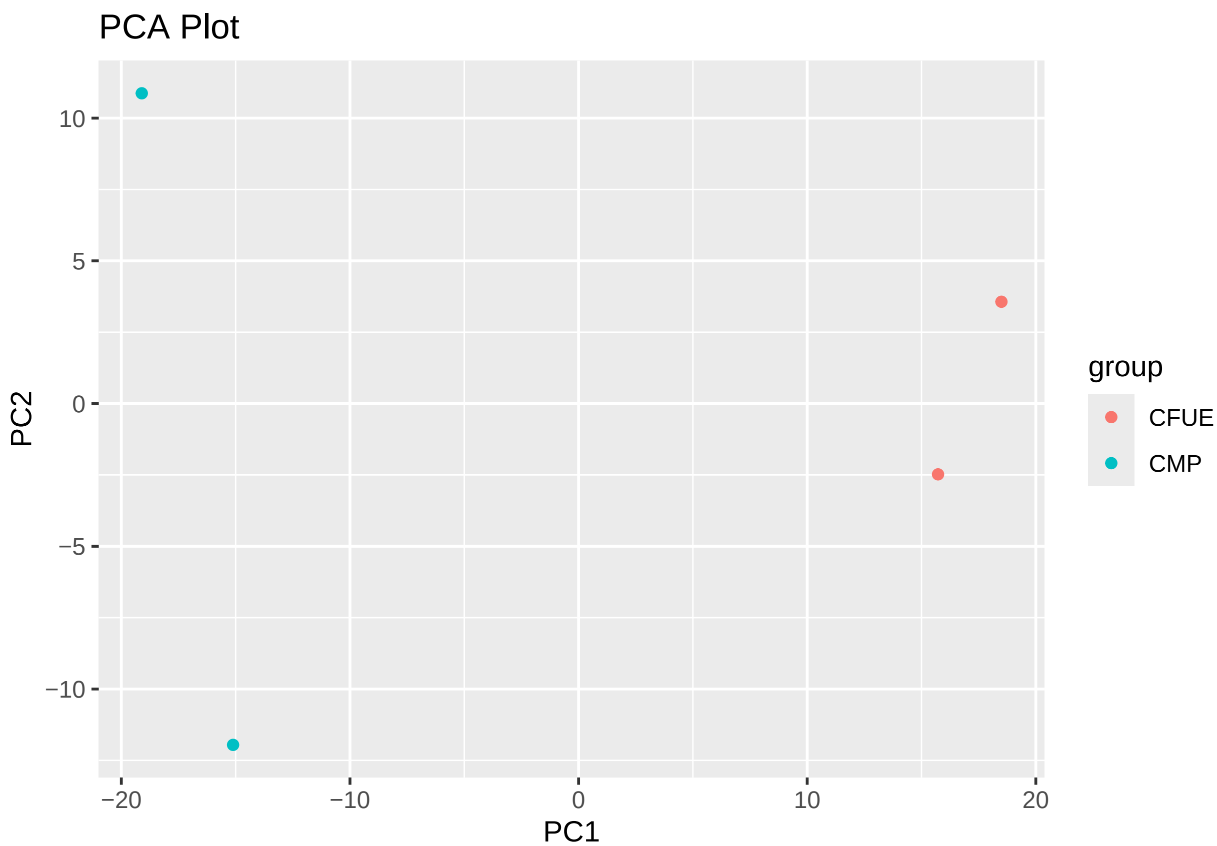
```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(ellipse)
```

```
##
## Attaching package: 'ellipse'

## The following object is masked from 'package:graphics':
##
##     pairs
```

```
M <- t(tpm[selectedGenes,])
M <- log2(M + 1)
pcaResults <- prcomp(M)

# Extract principal components from the PCA results
```

```r
pc <- data.frame(PC1 = pcaResults$x[,1], PC2 = pcaResults$x[,2])

# Add sample metadata from colData
pc <- cbind(pc, colData)

# Plot PCA results using ggplot2
ggplot(pc, aes(x = PC1, y = PC2, color = group)) +
  geom_point() +
  labs(title = "PCA Plot", x = "PC1", y = "PC2")
```




```r
summary(pcaResults)
```

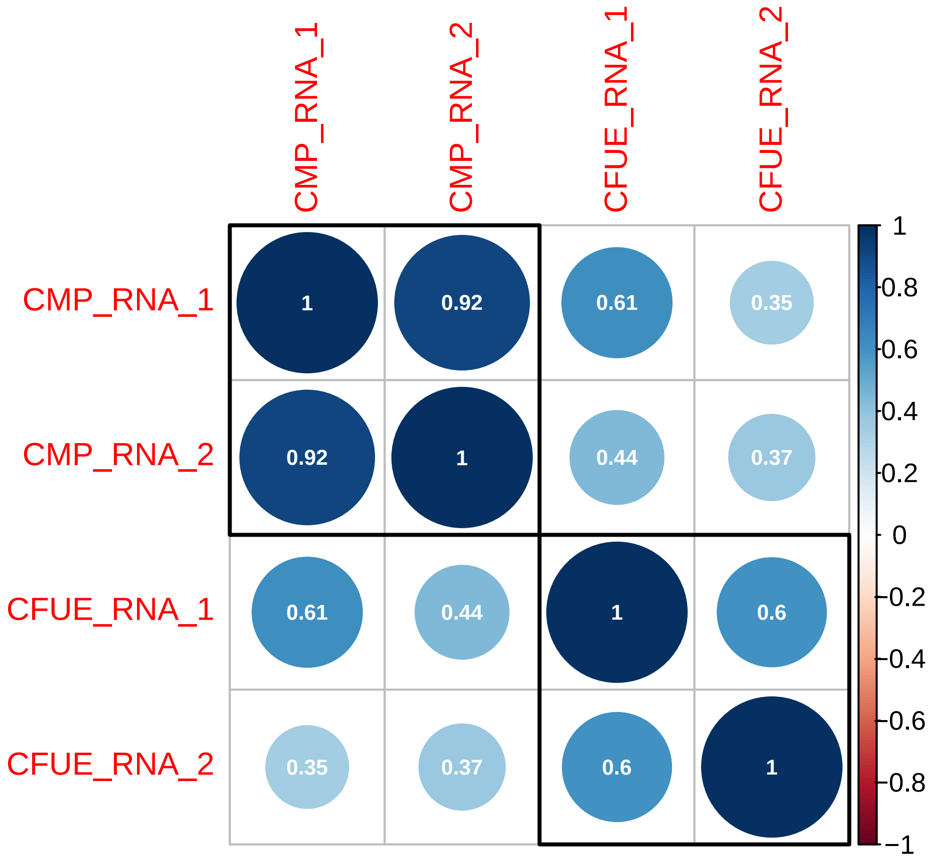
```
## Importance of components:
##                            PC1    PC2     PC3       PC4
## Standard deviation     19.8561 9.6610 5.65563 7.318e-15
## Proportion of Variance  0.7588 0.1796 0.06156 0.000e+00
## Cumulative Proportion   0.7588 0.9384 1.00000 1.000e+00
```

Correlation plots

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
correlationMatrix <- cor(tpm)
corrplot(correlationMatrix, order = 'hclust',
         addrect = 2, addCoef.col = 'white',
         number.cex = 0.7)
```



Differential Expression Analysis

Preprocessing

```r
#Get counts data
# Add prefixes to the tpm columns in each dataframe
CFUE_RNA_1_counts <- CFUE_RNA_1 %>% select(gene_id, CFUE_RNA_1_counts = expected_count)
CFUE_RNA_2_counts <- CFUE_RNA_2 %>% select(gene_id, CFUE_RNA_2_counts = expected_count)
CMP_RNA_1_counts <- CMP_RNA_1 %>% select(gene_id, CMP_RNA_1_counts = expected_count)
CMP_RNA_2_counts <- CMP_RNA_2 %>% select(gene_id, CMP_RNA_2_counts = expected_count)

# Join the dataframes together on gene_id
countData <- inner_join(CFUE_RNA_1_counts, CFUE_RNA_2_counts, by = "gene_id") %>%
       inner_join(CMP_RNA_1_counts, by = "gene_id") %>%
       inner_join(CMP_RNA_2_counts, by = "gene_id")

rm(CFUE_RNA_1_counts, CFUE_RNA_2_counts, CMP_RNA_1_counts, CMP_RNA_2_counts)
```

```r
countData <- countData %>%
  rename_at(vars(2:5), ~ sub("_counts$", "", .))

countData <- mutate_if(countData, is.numeric, round)

# Set 'column_name' as row names
rownames(countData) <- countData$gene_id

# Remove 'column_name' from dataframe (optional)
countData <- countData[, -which(names(countData) == 'gene_id')]

#Create a colData matrix
colData = data.frame(
  source_name = c('CMP_RNA_1','CMP_RNA_2','CFUE_RNA_1','CFUE_RNA_2'),
  group = c('CMP', 'CMP', 'CFUE', 'CFUE')
  )
colData$group = as.factor(colData$group)
designFormula <- "~ group"
```

DESeq2

```r
library(DESeq2)
```

```
## Warning: package 'DESeq2' was built under R version 4.3.3

## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

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

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

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, 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:dplyr':
##
##     first, rename

## 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 objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

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

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Warning: package 'GenomeInfoDb' was built under R version 4.3.3

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

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

##
## 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
```

```r
#create a DESeq dataset object from the count matrix and the colData
dds <- DESeqDataSetFromMatrix(countData = countData,
                              colData = colData,
                              design = as.formula(designFormula))
```

```
## converting counts to integer mode
```

```r
#print dds object to see the contents
print(dds)
```

```
## class: DESeqDataSet
## dim: 69691 4
## metadata(1): version
## assays(1): counts
```

```
## rowNames(69691): 10000 10001 ... gSpikein_ERCC-00171 gSpikein_phiX174
## rowData names(0):
## colnames(4): CFUE_RNA_1 CFUE_RNA_2 CMP_RNA_1 CMP_RNA_2
## colData names(2): source_name group
```

```
#For each gene, we count the total number of reads for that gene in all samples
#and remove those that don't have at least 1 read.
dds <- dds[ rowSums(DESeq2::counts(dds)) > 1, ]

dds <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
#compute the contrast for the 'group' variable where 'CTRL'
#samples are used as the control group.
DEresults = results(dds, contrast = c("group", 'CMP', 'CFUE'))
#sort results by increasing p-value
DEresults_print <- DEresults[order(DEresults$pvalue)[1:10],]
print(DEresults)
```

```
## log2 fold change (MLE): group CMP vs CFUE
## Wald test p-value: group CMP vs CFUE
## DataFrame with 18135 rows and 6 columns
##                          baseMean log2FoldChange     lfcSE       stat
##                         <numeric>      <numeric> <numeric>  <numeric>
## 22050                     5.16515        6.75602  4.804424    1.40621
## 31383                     9.18249        7.58864  4.795805    1.58235
## 46219                     6.20028       -5.21542  4.789963   -1.08882
## ENSMUSG00000000001.4   3656.99476       -1.12957  0.211447   -5.34209
## ENSMUSG00000000028.10  2398.80485        1.06024  0.205558    5.15787
## ...                           ...            ...       ...        ...
## ENSMUSG00000104514.1      5.94361     -0.0717867   3.73001 -0.0192457
## ENSMUSG00000104517.1      9.70745     -5.8571112   3.90893 -1.4983941
## ENSMUSG00000104523.1      2.86953      5.9034231   4.82020  1.2247248
## ENSMUSG00000104524.1     10.23288     -5.9366056   3.84766 -1.5429118
## ENSMUSG00000104525.1     28.70749      1.1467272   2.62809  0.4363354
##                            pvalue        padj
##                         <numeric>   <numeric>
## 22050                 1.59662e-01          NA
## 31383                 1.13570e-01 1.88324e-01
## 46219                 2.76232e-01          NA
```

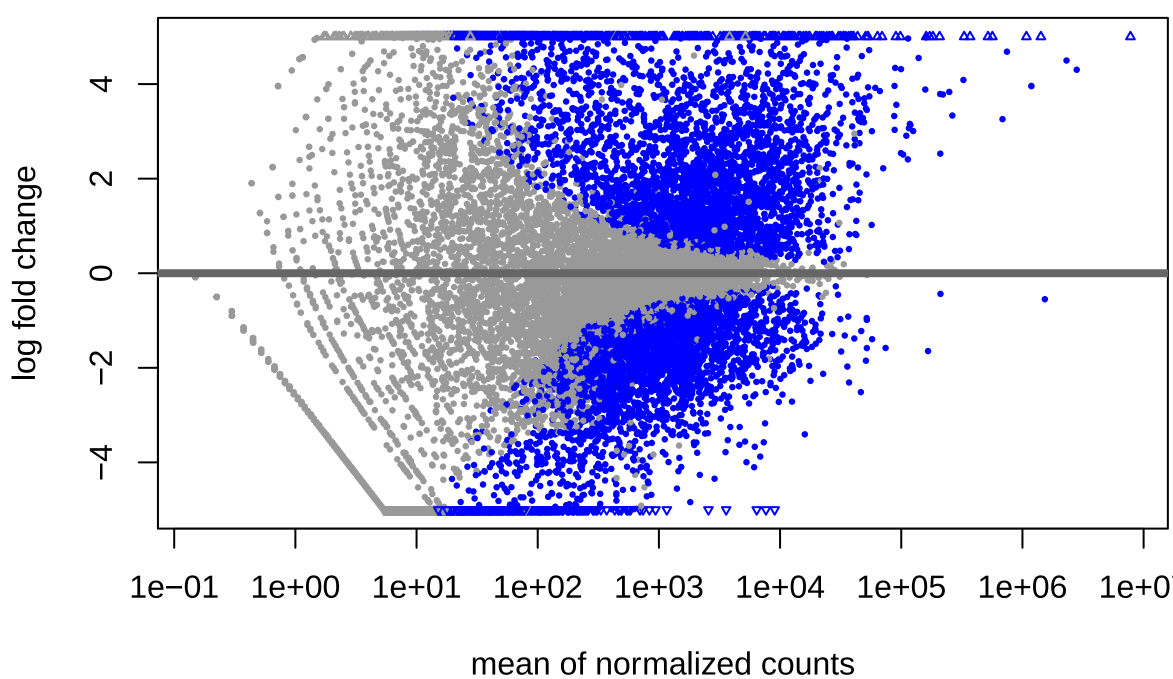
```
## ENSMUSG00000000001.4  9.18807e-08 5.19290e-07
## ENSMUSG00000000028.10 2.49779e-07 1.33909e-06
## ...                           ...         ...
## ENSMUSG00000104514.1     0.984645          NA
## ENSMUSG00000104517.1     0.134031    0.214097
## ENSMUSG00000104523.1     0.220679          NA
## ENSMUSG00000104524.1     0.122852    0.200183
## ENSMUSG00000104525.1     0.662593    0.749143
```

Diagnostic plots

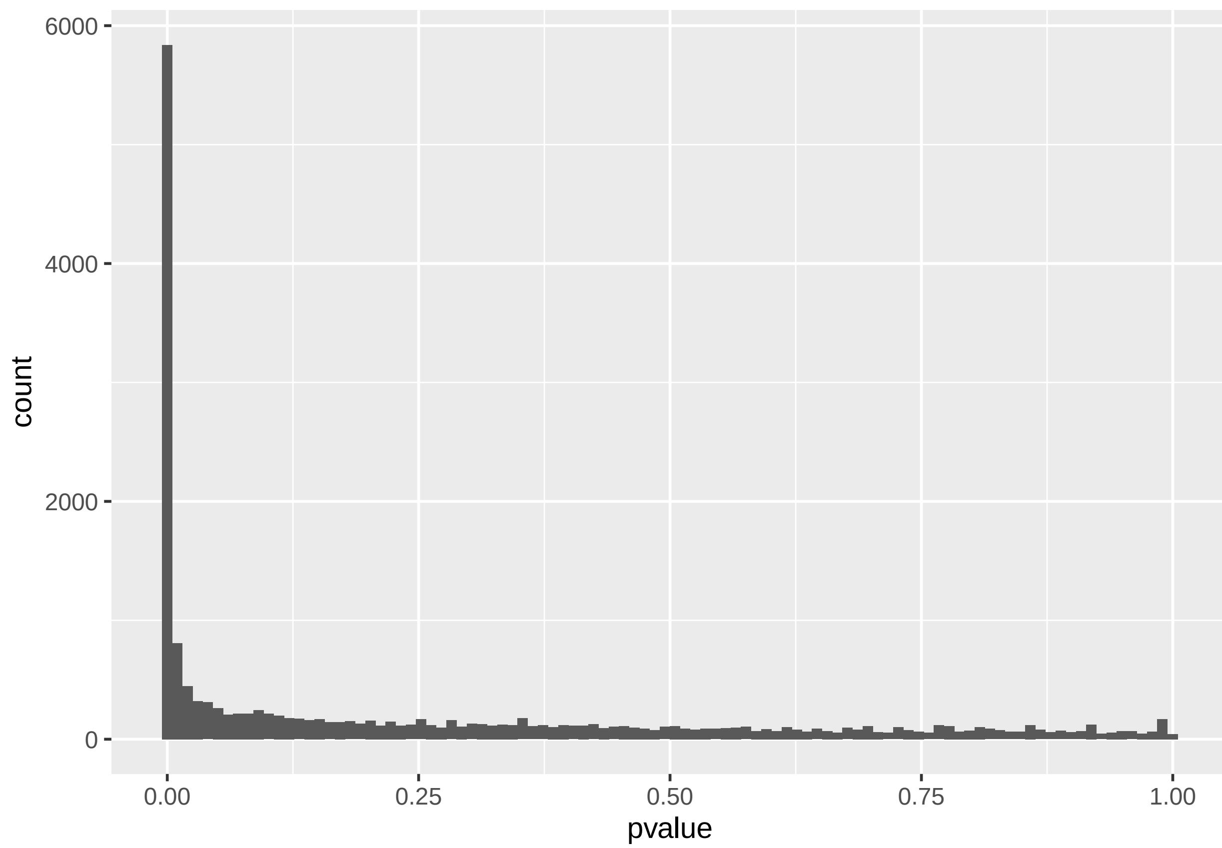
MA plot: Note that most points fall on the horizontal zero line, which means most genes are not differentially expressed.

P-value plot: It is also important to observe the distribution of raw p-values. We expect to see a peak around low p-values and a uniform distribution at P-values above 0.1. Otherwise, adjustment for multiple testing does not work and the results are not meaningful.

```
#MA plot
DESeq2::plotMA(object = dds, ylim = c(-5, 5))
```

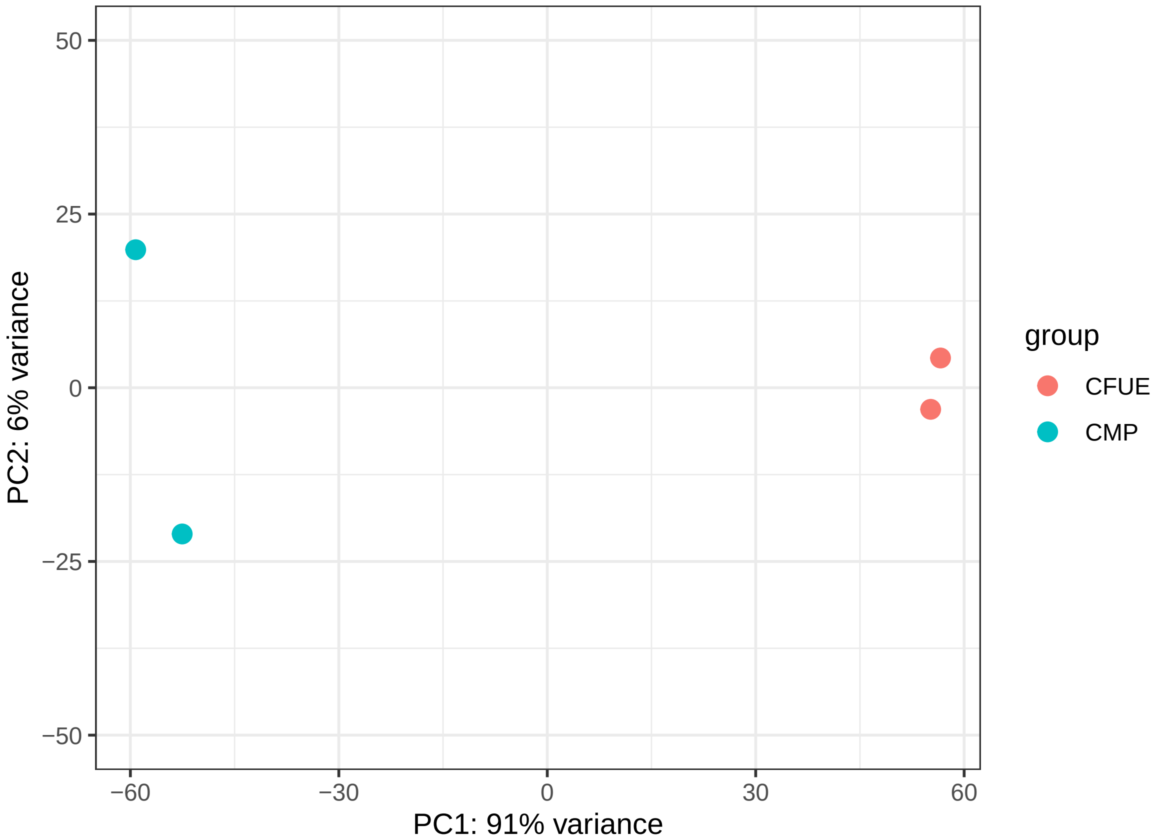



```
#Pvalue plot
library(ggplot2)
ggplot(data = as.data.frame(DEresults), aes(x = pvalue)) +
  geom_histogram(bins = 100)
```




```r
#PCA plot
rld <- rlog(dds)
DESeq2::plotPCA(rld, ntop = 500, intgroup = 'group') +
  ylim(-50, 50) + theme_bw()
```

```
## using ntop=500 top features by variance
```

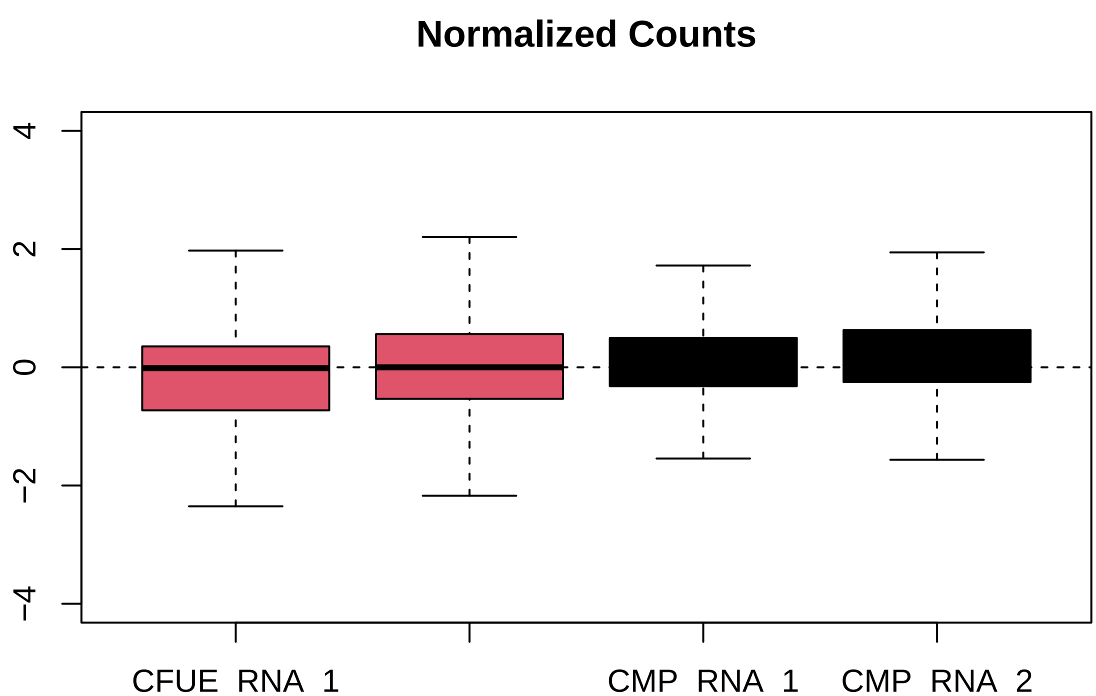



```
#RLE Plot
library(EDASeq)
```

```
## Loading required package: ShortRead

## Loading required package: BiocParallel

## Loading required package: Biostrings

## Warning: package 'Biostrings' was built under R version 4.3.3

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

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

## Loading required package: Rsamtools

## Loading required package: GenomicAlignments
```

```
##
## Attaching package: 'GenomicAlignments'

## The following object is masked from 'package:dplyr':
##
##     last

##
## Attaching package: 'ShortRead'

## The following object is masked from 'package:dplyr':
##
##     id
```

```r
par(mfrow = c(1, 1))
plotRLE(DESeq2::counts(dds, normalized = TRUE),
        outline=FALSE, ylim=c(-4, 4),
        col = as.numeric(colData$group),
        main = 'Normalized Counts')
```



Limma-VOOM

```r
library(edgeR)
```

```
## Loading required package: limma

##
## Attaching package: 'limma'

## The following object is masked from 'package:DESeq2':
##
##     plotMA

## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

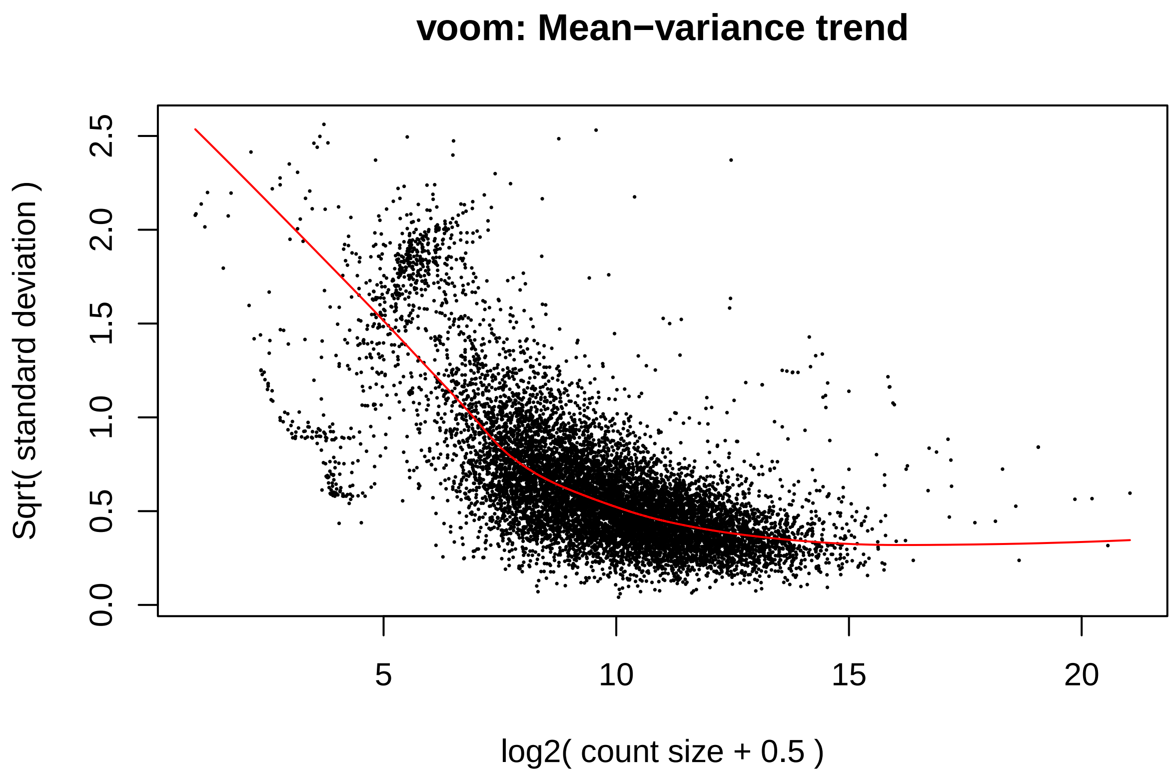
```r
#Create DGEList object
d0 <- DGEList(countData)

#Add normalizing factors
d0 <- calcNormFactors(d0)

#Drop low-expressed genes
cutoff <- 5
drop <- which(apply(cpm(d0), 1, max) < cutoff)
d <- d0[-drop,]
dim(d) # number of genes left
```

```
## [1] 10461     4
```

```r
group = c('CMP', 'CMP', 'CFUE', 'CFUE')

mm <- model.matrix(~group)
colnames(mm) <- gsub("group", "", colnames(mm))
print(mm)
```

```
##   (Intercept) CMP
## 1           1   1
## 2           1   1
## 3           1   0
## 4           1   0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

```r
y <- voom(d, mm, plot = T)
```




```r
fit <- lmFit(y, mm)
head(coef(fit))
```

```
##                         (Intercept)        CMP
## ENSMUSG00000000001.4       6.766653 -1.2259044
## ENSMUSG00000000028.10      5.085925  0.9858912
## ENSMUSG00000000037.12      2.916188 -1.8348093
## ENSMUSG00000000056.7       5.084557  3.1718361
## ENSMUSG00000000078.6       6.892559 -1.3288026
## ENSMUSG00000000085.12      3.009781  0.3878810
```

```r
tmp <- eBayes(fit)

top.table <- topTable(tmp, sort.by = "P", n = Inf)
```

```
## Removing intercept from test coefficients
```

```r
lv_dif_ex_genes = rownames(head(top.table, 1000))
```

```r
library(VennDiagram)
```

```
## Warning: package 'VennDiagram' was built under R version 4.3.3
```

```
## Loading required package: grid
```

```
##
## Attaching package: 'grid'

## The following object is masked from 'package:Biostrings':
##
##     pattern

## Loading required package: futile.logger

## Warning: package 'futile.logger' was built under R version 4.3.3

##
## Attaching package: 'VennDiagram'

## The following object is masked from 'package:ellipse':
##
##     ellipse
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

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

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
# extract differential expression results
DEresults <- results(dds, contrast = c('group', 'CMP', 'CFUE'))

#remove genes with NA values
DE <- DEresults[!is.na(DEresults$padj),]
lowest_pvalue_indexes <- order(DE@listData$pvalue)[1:1000]
DE2_dif_ex_genes = DE@rownames[lowest_pvalue_indexes]

# Create a Venn diagram
venn.plot <- venn.diagram(
  x = list(lv_dif_ex_genes, DE2_dif_ex_genes),
  category.names = c("Limma-voom" , "DESEQ2"),
  filename = '#venn_diagramm.png',
  output=TRUE
)
```

GO term analysis

Simply run this code on a set of genes that are significantly upregulated and then same with downregulated

```r
library(DESeq2)
library(gprofiler2)
```

Find the upregulated and downregulated gene descriptions

```
## Warning: package 'gprofiler2' was built under R version 4.3.3
```

```r
library(knitr)
# extract differential expression results
DEresults <- results(dds, contrast = c('group', 'CMP', 'CFUE'))

#remove genes with NA values
DE <- DEresults[!is.na(DEresults$padj),]
#select genes with adjusted p-values below 0.1
DE <- DE[DE$padj < 0.1,]
#select genes with log2 fold change above 1 (two-fold change)
up_reg_DE <- DE[DE$log2FoldChange > 1,]
dn_reg_DE <- DE[DE$log2FoldChange < -1,]

#get the list of upregulated genes of interest
up_reg_gene_names <- rownames(up_reg_DE)
up_reg_gene_names <- sapply(strsplit(up_reg_gene_names, "\\."), function(x) x[[1]])
up_reg_gene_names <- unique(up_reg_gene_names)

#get the list of downregulated genes of interest
dn_reg_gene_names <- rownames(dn_reg_DE)
dn_reg_gene_names <- sapply(strsplit(dn_reg_gene_names, "\\."), function(x) x[[1]])
dn_reg_gene_names <- unique(dn_reg_gene_names)

#calculate enriched GO terms
up_go_response <- gost(query = up_reg_gene_names,
                       organism = 'mmusculus',
                    sources = c("GO"))
dn_go_response <- gost(query = dn_reg_gene_names,
                       organism = 'mmusculus',
                    sources = c("GO"))

# gostplot(up_go_response, capped=FALSE)
up_go_results = up_go_response$result
dn_go_results = dn_go_response$result
up_go_results <- up_go_results[order(up_go_results$p_value),]
dn_go_results <- dn_go_results[order(dn_go_results$p_value),]
# up_go_results <- up_go_results[up_go_results$intersection_size < 100,]

kable(up_go_results[1:10,c(7:11)])
```

	precision	recall	term_id	source	term_name
864	0.6466780	0.1641726	GO: 0005737	GO:CC	cytoplasm
1025	0.6250000	0.1720841	GO: 0005515	GO: MF	protein binding
865	0.8293015	0.1397887	GO: 0005622	GO:CC	intracellular anatomical structure
866	0.9550256	0.1237364	GO: 0110165	GO:CC	cellular anatomical entity
867	0.7570698	0.1403132	GO: 0043229	GO:CC	intracellular organelle
868	0.7642249	0.1387222	GO: 0043226	GO:CC	organelle
1026	0.8121528	0.1428135	GO: 0005488	GO: MF	binding
1	0.4013135	0.1762028	GO: 0048518	GO:BP	positive regulation of biological process
869	0.7001704	0.1423623	GO: 0043231	GO:CC	intracellular membrane-bounded organelle
870	0.7148211	0.1404848	GO: 0043227	GO:CC	membrane-bounded organelle

```
kable(dn_go_results[1:10,c(7:11)])
```

	precision	recall	term_id	source	term_name
569	0.7038184	0.1769743	GO: 0005737	GO: CC	cytoplasm
570	0.8654971	0.1444980	GO: 0005622	GO: CC	intracellular anatomical structure
571	0.7997936	0.1437937	GO: 0043226	GO: CC	organelle
572	0.7884417	0.1447335	GO: 0043229	GO: CC	intracellular organelle
573	0.7481940	0.1456408	GO: 0043227	GO: CC	membrane-bounded organelle
574	0.7230822	0.1456183	GO: 0043231	GO: CC	intracellular membrane-bounded organelle
1	0.3296590	0.2029692	GO: 0044271	GO: BP	cellular nitrogen compound biosynthetic process
2	0.0657940	0.5950156	GO: 0042254	GO: BP	ribosome biogenesis
575	0.2948056	0.2087698	GO: 0005829	GO: CC	cytosol
576	0.9552804	0.1225886	GO: 0110165	GO: CC	cellular anatomical entity

```
# gostplot(up_go_response, capped = FALSE)
# gostplot(dn_go_response, capped = FALSE)
```