Deseq2 analysis 0dpi

#KEY INFO

#PIPELINE OVERVIEW  
#https://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html  
  
#https://master.bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html  
  
#https://github.com/thelovelab/DESeq2/tree/devel/R   
  
#INFO ON UPLOADING DATA WITH MULTIPLE CONDITIONS  
  
#https://bioinformatics-core-shared-training.github.io/cruk-summer-school-2018/RNASeq2018/html/04\_DE\_analysis\_with\_DESeq2.nb.html   
  
#https://rstudio-pubs-static.s3.amazonaws.com/329027\_593046fb6d7a427da6b2c538caf601e1.html

#LOAD PACKAGES

library('BiocManager') #for installing bioconductor packages like DESeq2  
library('DESeq2') #for analyzing differential gene expression  
library("mixOmics") #for PCA and PLSDA  
library('ggplot2') #for plotting  
library("VennDiagram") #for venn diagrams  
library("ComplexHeatmap") #for heatmaps  
library("viridis") #for gene clustering heatmap colours  
library('RColorBrewer') #for sample-sample heatmap colours  
library('dendextend') #for editing dendrograms in gene clustering heatmap  
library("dplyr") #for data manipulation  
library('caret') #for removing zero variance data for PCA/PLSDA  
library('seqinr') #for reading fasta files into R

#LOAD 0DPI DATA

#read in sample info file  
#note have combined mock and different elicitors into single "condition column" as only comparing effect of elicitors   
zero\_dpi\_sample\_info <- read.table("raw\_count/0dpi/sample\_info\_0dpi.csv", header = TRUE, sep = ",")  
  
#add rownames to sample info = for labelling of plots (e.g., heatmaps)  
row.names(zero\_dpi\_sample\_info) <- zero\_dpi\_sample\_info$sample\_name  
  
#create factors for condition column so that water will be used as intercept/base-line/reference for the model  
zero\_dpi\_sample\_info$condition <- factor(zero\_dpi\_sample\_info$condition, levels = c("Mock\_Water", "Mock\_BABA", "Mock\_JA", "Mock\_SA"))  
  
#visualize the model matrix that will be used by Deseq2  
model.matrix(~condition, data = zero\_dpi\_sample\_info)

## (Intercept) conditionMock\_BABA conditionMock\_JA conditionMock\_SA  
## MW0\_1 1 0 0 0  
## MW0\_2 1 0 0 0  
## MW0\_3 1 0 0 0  
## MB0\_1 1 1 0 0  
## MB0\_2 1 1 0 0  
## MB0\_3 1 1 0 0  
## MB0\_4 1 1 0 0  
## MJ0\_1 1 0 1 0  
## MJ0\_2 1 0 1 0  
## MJ0\_3 1 0 1 0  
## MJ0\_4 1 0 1 0  
## MS0\_1 1 0 0 1  
## MS0\_2 1 0 0 1  
## MS0\_3 1 0 0 1  
## MS0\_4 1 0 0 1  
## attr(,"assign")  
## [1] 0 1 1 1  
## attr(,"contrasts")  
## attr(,"contrasts")$condition  
## [1] "contr.treatment"

#should display 4 columns - intercept (MW), MB, MJA, MSA  
  
#read in gene counts data  
zero\_dpi\_raw\_gene\_data <- DESeqDataSetFromHTSeqCount(  
 sampleTable = zero\_dpi\_sample\_info, directory = 'raw\_count/0dpi',  
 design = ~ condition)

#QUALITY SCATTERPLOTS

#extract and save normalised counts  
zero\_dpi\_raw\_gene\_data <- estimateSizeFactors(zero\_dpi\_raw\_gene\_data)  
zero\_dpi\_normalised\_raw\_gene\_data <- counts(zero\_dpi\_raw\_gene\_data, normalized=TRUE)  
  
#set up pdf to save graphs  
quality\_scatterplots <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_quality\_scatterplots.pdf"  
pdf(file = quality\_scatterplots)  
  
#water  
par(mfrow = c(2,3)) #2 rows, 3 cols  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,1]), log(zero\_dpi\_normalised\_raw\_gene\_data[,2]), xlab = "0MW\_1", ylab = "0MW\_2")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "A")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,1]), log(zero\_dpi\_normalised\_raw\_gene\_data[,3]), xlab = "0MW\_1", ylab = "0MW\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "B")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,2]), log(zero\_dpi\_normalised\_raw\_gene\_data[,3]), xlab = "0MW\_2", ylab = "0MW\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "C")  
  
#BABA  
par(mfrow = c(2,3)) #2 rows, 3 cols  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,4]), log(zero\_dpi\_normalised\_raw\_gene\_data[,5]), xlab = "0MB\_1", ylab = "0MB\_2")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "A")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,4]), log(zero\_dpi\_normalised\_raw\_gene\_data[,6]), xlab = "0MB\_1", ylab = "0MB\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "B")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,4]), log(zero\_dpi\_normalised\_raw\_gene\_data[,7]), xlab = "0MB\_1", ylab = "0MB\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "C")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,5]), log(zero\_dpi\_normalised\_raw\_gene\_data[,6]), xlab = "0MB\_2", ylab = "0MB\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "D")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,5]), log(zero\_dpi\_normalised\_raw\_gene\_data[,7]), xlab = "0MB\_2", ylab = "0MB\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "E")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,6]), log(zero\_dpi\_normalised\_raw\_gene\_data[,7]), xlab = "0MB\_3", ylab = "0MB\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "F")  
  
#JA  
par(mfrow = c(2,3)) #2 rows, 3 cols  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,8]), log(zero\_dpi\_normalised\_raw\_gene\_data[,9]), xlab = "0MJ\_1", ylab = "0MJ\_2")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "A")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,8]), log(zero\_dpi\_normalised\_raw\_gene\_data[,10]), xlab = "0MJ\_1", ylab = "0MJ\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "B")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,8]), log(zero\_dpi\_normalised\_raw\_gene\_data[,11]), xlab = "0MJ\_1", ylab = "0MJ\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "C")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,9]), log(zero\_dpi\_normalised\_raw\_gene\_data[,10]), xlab = "0MJ\_2", ylab = "0MJ\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "D")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,9]), log(zero\_dpi\_normalised\_raw\_gene\_data[,11]), xlab = "0MJ\_2", ylab = "0MJ\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "E")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,10]), log(zero\_dpi\_normalised\_raw\_gene\_data[,11]), xlab = "0MJ\_3", ylab = "0MJ\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "F")  
  
#SA  
par(mfrow = c(2,3)) #2 rows, 3 cols  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,12]), log(zero\_dpi\_normalised\_raw\_gene\_data[,13]), xlab = "0MS\_1", ylab = "0MS\_2")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "A")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,12]), log(zero\_dpi\_normalised\_raw\_gene\_data[,14]), xlab = "0MS\_1", ylab = "0MS\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "B")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,12]), log(zero\_dpi\_normalised\_raw\_gene\_data[,15]), xlab = "0MS\_1", ylab = "0MS\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "C")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,13]), log(zero\_dpi\_normalised\_raw\_gene\_data[,14]), xlab = "0MS\_2", ylab = "0MS\_3")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "D")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,13]), log(zero\_dpi\_normalised\_raw\_gene\_data[,15]), xlab = "0MS\_2", ylab = "0MS\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "E")  
  
plot(log(zero\_dpi\_normalised\_raw\_gene\_data[,14]), log(zero\_dpi\_normalised\_raw\_gene\_data[,15]), xlab = "0MS\_3", ylab = "0MS\_4")  
  
abline(a = 0, b = 1, col = "red")  
  
title(main = "F")  
  
#save all graphs  
dev.off()

#TRANSFORMATIONS

#two transformation used in DESeq2...  
  
vst\_zero\_dpi\_raw\_gene\_data <- vst(zero\_dpi\_raw\_gene\_data) #vst transformation, recommend the VST for medium-to-large datasets (n > 30), see Anders, Simon, and Wolfgang Huber. 2010. “Differential expression analysis for sequence count data.” Genome Biology 11 (10): R106+. https://doi.org/10.1186/gb-2010-11-10-r106.   
  
rlog\_zero\_dpi\_raw\_gene\_data <- rlog(zero\_dpi\_raw\_gene\_data) #rlog transformation, the rlog tends to work well on small datasets (n < 30), potentially outperforming the VST when there is a wide range of sequencing depth across samples (an order of magnitude difference), see Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. “Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.” Genome Biology 15 (12): 550+. https://doi.org/10.1186/s13059-014-0550-8.  
  
  
#plot comparison with conventional log transformation (not recommended but shown for comparison)  
zero\_dpi\_raw\_gene\_data <- estimateSizeFactors(zero\_dpi\_raw\_gene\_data) #done so log transform can account for sequencing depth (this is automatic for vst and rlog)  
  
transformation\_df <- bind\_rows(  
 as.data.frame(log2(counts(zero\_dpi\_raw\_gene\_data, normalized=TRUE)[, 1:2]+1)) %>%  
 mutate(transformation = "log2(x + 1)"),  
 as.data.frame(assay(vst\_zero\_dpi\_raw\_gene\_data)[, 1:2]) %>% mutate(transformation = "vst"),  
 as.data.frame(assay(rlog\_zero\_dpi\_raw\_gene\_data)[, 1:2]) %>% mutate(transformation = "rlog"))  
   
colnames(transformation\_df)[1:2] <- c("x", "y")   
  
lvls <- c("log2(x + 1)", "vst", "rlog")  
transformation\_df$transformation <- factor(transformation\_df$transformation, levels=lvls)  
  
p <- ggplot(transformation\_df, aes(x = x, y = y)) + geom\_hex(bins = 80) +  
 coord\_fixed() + facet\_grid( . ~ transformation)   
  
ggsave(filename = "example\_zero\_dpi\_transformations.pdf", plot = p, width = 15, height = 9, path = "Deseq\_analysis", device = "pdf")  
  
# KEY POINT = We can see how genes with low counts (bottom left-hand corner) seem to be excessively variable on the ordinary logarithmic scale, while the VST and rlog compress differences for the low count genes for which the data provide little information about differential expression.

#SAMPLE-SAMPLE DISTANCE HEATMAP

#Basically shows how similar the different samples are i.e. sample clustering, uses normalised data  
  
#set up pdf to save graphs  
sample\_clustering\_dend <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_sample\_clustering\_dend.pdf"  
pdf(file = sample\_clustering\_dend, width = 5, height = 5)  
  
#first generate sample-sample distance matrix  
sampleDistances <- dist(t(assay(rlog\_zero\_dpi\_raw\_gene\_data))) #uses euclidean distances as default  
sampleDistMatrix <- as.matrix(sampleDistances)  
colnames(sampleDistMatrix) <- rownames(zero\_dpi\_sample\_info)  
  
#perform hierachical clustering and plot dendrogram  
col = c("blue", "blue", "blue", "#117733", "#117733", "#117733", "#AA4499", "#AA4499", "#AA4499", "#AA4499", "#DDCC77", "#DDCC77", "#DDCC77", "#117733", "#DDCC77") #set up sample colours  
zero\_dpi\_sample\_dendrogram <- as.dendrogram(hclust(sampleDistances)) #cluster using complete linkage method as default  
dend <- dendextend::set(zero\_dpi\_sample\_dendrogram, "labels\_colors", col) #color code labels  
plot(dend) #plot  
dendextend::rect.dendrogram(dend, k = 5, border = "red", lty = "dashed") #add dashed red rectangles around major clusters (visually there are ~5)  
  
dev.off()

sample\_clustering <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_sample\_clustering.pdf"  
pdf(file = sample\_clustering, width = 10, height = 10)  
  
#create colour scheme  
colours <- colorRampPalette(rev(brewer.pal(9, "Blues")))(255)  
  
#set up sample annotations for plot  
sample\_annotations <- zero\_dpi\_sample\_info[,3]  
  
treatments\_top <- HeatmapAnnotation(Condition = sample\_annotations,   
 col = list(Condition = c("Mock\_Water" = "blue", "Mock\_BABA" = "#117733", "Mock\_JA" = "#AA4499", "Mock\_SA" = "#DDCC77")),   
 show\_annotation\_name = F, gp = gpar(col = "black"))  
  
  
treatments\_left <- rowAnnotation(Condition = sample\_annotations,   
 col = list(Condition = c("Mock\_Water" = "blue", "Mock\_BABA" = "#117733", "Mock\_JA" = "#AA4499", "Mock\_SA" = "#DDCC77")),   
 show\_legend = F, show\_annotation\_name = F, gp = gpar(col = "black"))  
  
#plot heatmap  
ComplexHeatmap::Heatmap(sampleDistMatrix, clustering\_distance\_rows = sampleDistances, clustering\_distance\_columns = sampleDistances,  
 col = colours, top\_annotation = treatments\_top, left\_annotation = treatments\_left,  
 heatmap\_legend\_param = list(title = NULL, border = "black"),  
 show\_row\_names = F, show\_column\_names = F, column\_title = "0dpi Sample Clustering Heatmap")  
  
dev.off()

#SET UP PCA AND PLSDA

#HAVE CURRENTLY USED RLOG TRANSFOMATION BUT UNSURE IF SHOULD USE VST  
  
#extract rlog expression data (unsure if use rlog expression or normal expression)  
rlog\_expression\_zero\_dpi <- t(assay(rlog\_zero\_dpi\_raw\_gene\_data))  
dim(rlog\_expression\_zero\_dpi) #check dimensions of resulting matrix

## [1] 15 25808

#need to remove colums with all 0s/zero variance as otherwise PLS/PCA will throw error: columns with zero variance in 'X'  
all\_cols <- colnames(rlog\_expression\_zero\_dpi) #identify names of all cols in rlog\_expression\_zero\_dpi  
remove\_cols <- caret::nearZeroVar(rlog\_expression\_zero\_dpi, names = TRUE) #identify cols with zero var to remove in rlog\_expression\_zero\_dpi  
rlog\_expression\_zero\_dpi <- rlog\_expression\_zero\_dpi[ , setdiff(all\_cols, remove\_cols)] #remove cols with zero var from rlog\_expression\_zero\_dpi  
dim(rlog\_expression\_zero\_dpi) #check dimensions of resulting matrix, should be smaller

## [1] 15 21956

#extract classes (conditions = 0MW, 0MB, 0MS, 0MJ)  
rlog\_classes\_zero\_dpi <- colData(rlog\_zero\_dpi\_raw\_gene\_data)  
rlog\_classes\_zero\_dpi <- rlog\_classes\_zero\_dpi$condition  
  
#check that all conditions extracted as expected  
summary(rlog\_classes\_zero\_dpi)

## Mock\_Water Mock\_BABA Mock\_JA Mock\_SA   
## 3 4 4 4

#PCA SCORES

#As far as I can tell from MetaboanalystR page prcomp is used by Metaboanalyst for PCA  
#However I have used mixOmics for ease of use and consistency with plsda  
  
#of note is that scale = FALSE with prcomp, but mixOmics advises to set scale = TRUE (this lowers %variance explained slightly)  
  
#set up pdf to save graphs  
PCA <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_PCA\_scores\_plots.pdf"  
pdf(file = PCA)  
  
#create PCA model  
pca\_zero\_dpi <- pca(rlog\_expression\_zero\_dpi, ncomp = 14, center = TRUE, scale = TRUE) #used ncomp = 14 so numbers in scree plot below sum to 100%  
  
#plot pca for all samples comps 1 and 2  
plotIndiv(pca\_zero\_dpi, comp = c(1,2), group = rlog\_classes\_zero\_dpi, ind.names = FALSE, ellipse = TRUE, ellipse.level = 0.95, pch = 19, cex = 3, size.xlabel = rel(2), size.ylabel = rel(2), size.axis = rel(1.5),  
 legend = TRUE, title = '0dpi PCA: PC1 vs PC2', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))  
  
plotIndiv(pca\_zero\_dpi, comp = c(1,3), group = rlog\_classes\_zero\_dpi, ind.names = FALSE, ellipse = TRUE, ellipse.level = 0.95, pch = 19, cex = 3, size.xlabel = rel(2), size.ylabel = rel(2), size.axis = rel(1.5),  
 legend = TRUE, title = '0dpi PCA: PC1 vs PC3', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))   
  
plotIndiv(pca\_zero\_dpi, comp = c(2,3), group = rlog\_classes\_zero\_dpi, ind.names = FALSE, ellipse = TRUE, ellipse.level = 0.95, pch = 19, cex = 3, size.xlabel = rel(2), size.ylabel = rel(2), size.axis = rel(1.5),  
 legend = TRUE, title = '0dpi PCA: PC2 vs PC3', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))  
  
#from pca plots components 1 and 3 seem to give clearest separation but still not obvious, note lower %variance due to scale = TRUE  
  
dev.off()

#PCA LOADINGS

#set up pdf to save graphs  
pca\_loadings <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_PCA\_loadings.pdf"  
pdf(file = pca\_loadings)  
  
#extract loadings data  
zero\_dpi\_pca\_loadings <- as.data.frame(pca\_zero\_dpi$loadings)  
  
#PC1 vs PC2  
ggplot(data = zero\_dpi\_pca\_loadings)+  
 geom\_point(mapping = aes(x = X.PC1, y = X.PC2))+  
 scale\_y\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 scale\_x\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 xlab("Loadings 1")+  
 ylab("Loadings 2")+  
 ggtitle("0dpi loadings plot")  
  
#PC1 vs PC3  
ggplot(data = zero\_dpi\_pca\_loadings)+  
 geom\_point(mapping = aes(x = X.PC1, y = X.PC3))+  
 scale\_y\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 scale\_x\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 xlab("Loadings 1")+  
 ylab("Loadings 3")+  
 ggtitle("0dpi loadings plot")  
  
#PC2 vs PC3  
ggplot(data = zero\_dpi\_pca\_loadings)+  
 geom\_point(mapping = aes(x = X.PC2, y = X.PC3))+  
 scale\_y\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 scale\_x\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 xlab("Loadings 2")+  
 ylab("Loadings 3")+  
 ggtitle("0dpi loadings plot")  
  
dev.off()

#PCA SCREE PLOT

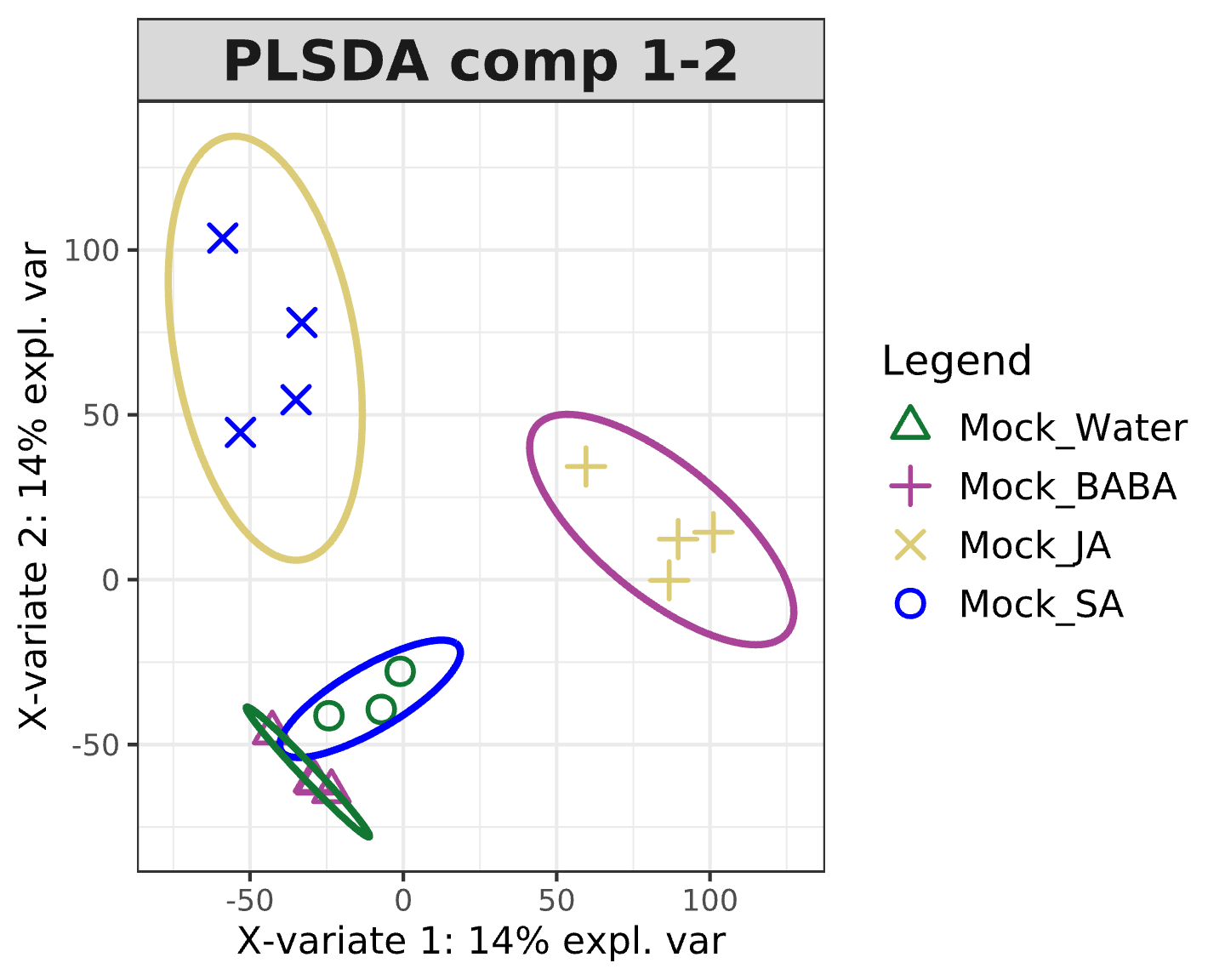
#set up and plot scree plot  
zero\_dpi\_expl\_var <- as.data.frame(pca\_zero\_dpi$prop\_expl\_var) #extract explained variance  
names(zero\_dpi\_expl\_var)[names(zero\_dpi\_expl\_var) == 'X'] <- 'percent\_var' #rename percent col  
zero\_dpi\_expl\_var <- cbind(PC\_index = 1:14, zero\_dpi\_expl\_var) #add PC\_index column  
zero\_dpi\_expl\_var$percent\_var <- zero\_dpi\_expl\_var$percent\_var \* 100 #convert decimal to percent  
zero\_dpi\_expl\_var$cum\_var <- cumsum(zero\_dpi\_expl\_var$percent\_var) #calculate cumulative variance explained  
zero\_dpi\_expl\_var$percent\_var <- round(zero\_dpi\_expl\_var$percent\_var, digits = 1) #round to 1dp  
zero\_dpi\_expl\_var$cum\_var <- round(zero\_dpi\_expl\_var$cum\_var, digits = 1)  
zero\_dpi\_expl\_var

## PC\_index percent\_var cum\_var  
## PC1 1 16.6 16.6  
## PC2 2 14.2 30.8  
## PC3 3 12.2 43.0  
## PC4 4 9.2 52.2  
## PC5 5 7.1 59.3  
## PC6 6 6.4 65.7  
## PC7 7 6.1 71.8  
## PC8 8 5.8 77.6  
## PC9 9 5.2 82.8  
## PC10 10 4.6 87.4  
## PC11 11 3.7 91.1  
## PC12 12 3.2 94.3  
## PC13 13 3.0 97.3  
## PC14 14 2.7 100.0

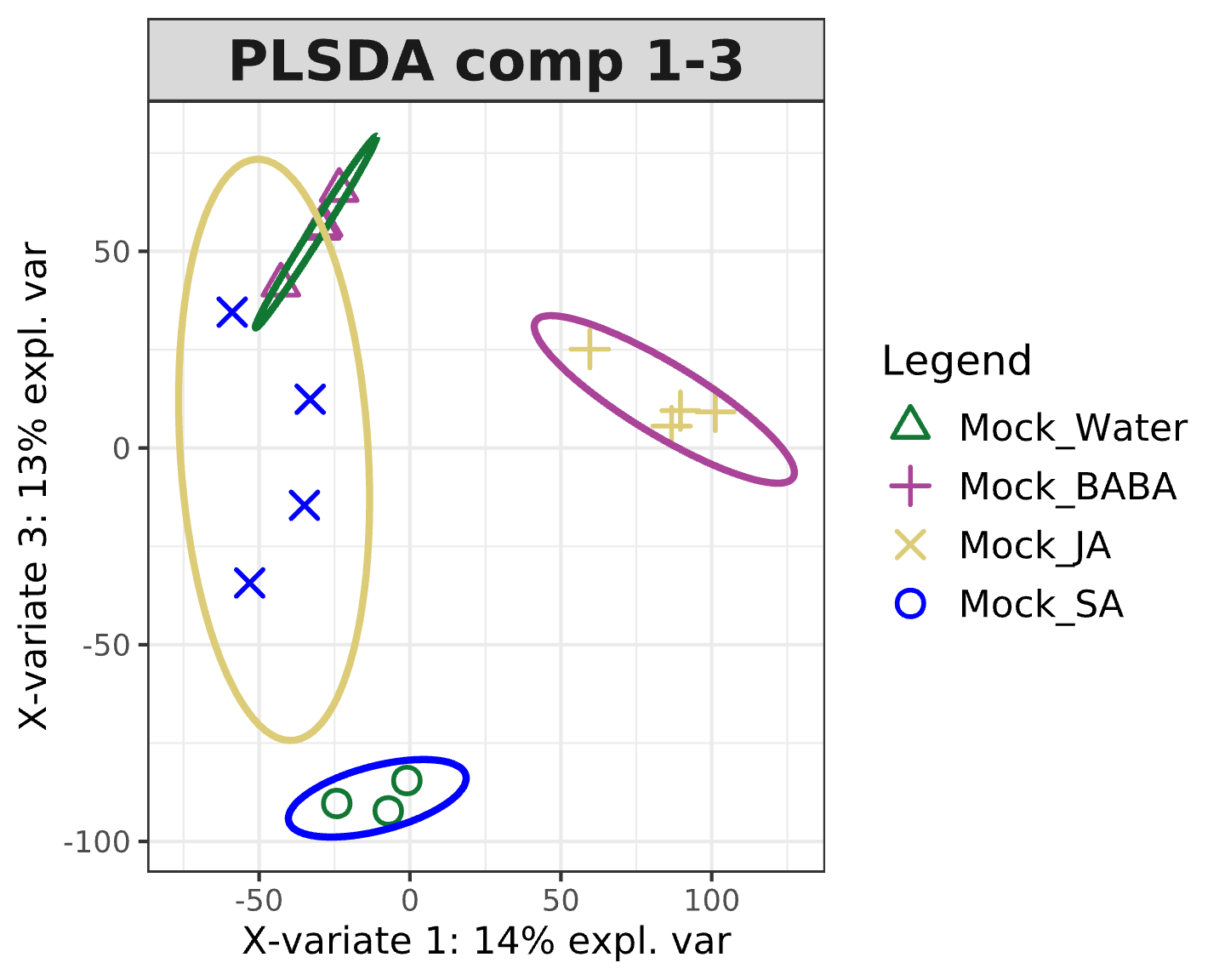
scree\_plot <- ggplot(data = zero\_dpi\_expl\_var)+  
 geom\_point(mapping = aes(x = PC\_index, y = percent\_var))+  
 geom\_line(mapping = aes(x = PC\_index, y = percent\_var), col = "blue")+  
 geom\_text(aes(x=PC\_index, y=percent\_var, label=percent\_var), nudge\_y = 4)+  
 geom\_point(mapping = aes(x = PC\_index, y = cum\_var))+  
 geom\_line(mapping = aes(x = PC\_index, y = cum\_var), col = "#117733")+  
 geom\_text(aes(x=PC\_index, y=cum\_var, label=cum\_var), nudge\_y = 4)+  
 scale\_x\_continuous(breaks = seq(0,15, by = 1))+  
 xlab("PC index")+  
 ylab("Percentage of variance explained (%)")+  
 theme(panel.grid.major.y = element\_blank(), panel.grid.minor.y = element\_blank(), panel.grid.minor.x = element\_blank(), panel.grid.major.x = element\_line(linetype = "dashed", colour = "grey"))+  
 ggtitle("0dpi Scree Plot")  
  
ggsave(filename = "zero\_dpi\_PCA\_scree\_plot.pdf", plot = scree\_plot, width = 15, height = 9, path = "Deseq\_analysis/zero\_dpi\_out", device = "pdf")

#INITIAL PLSDA

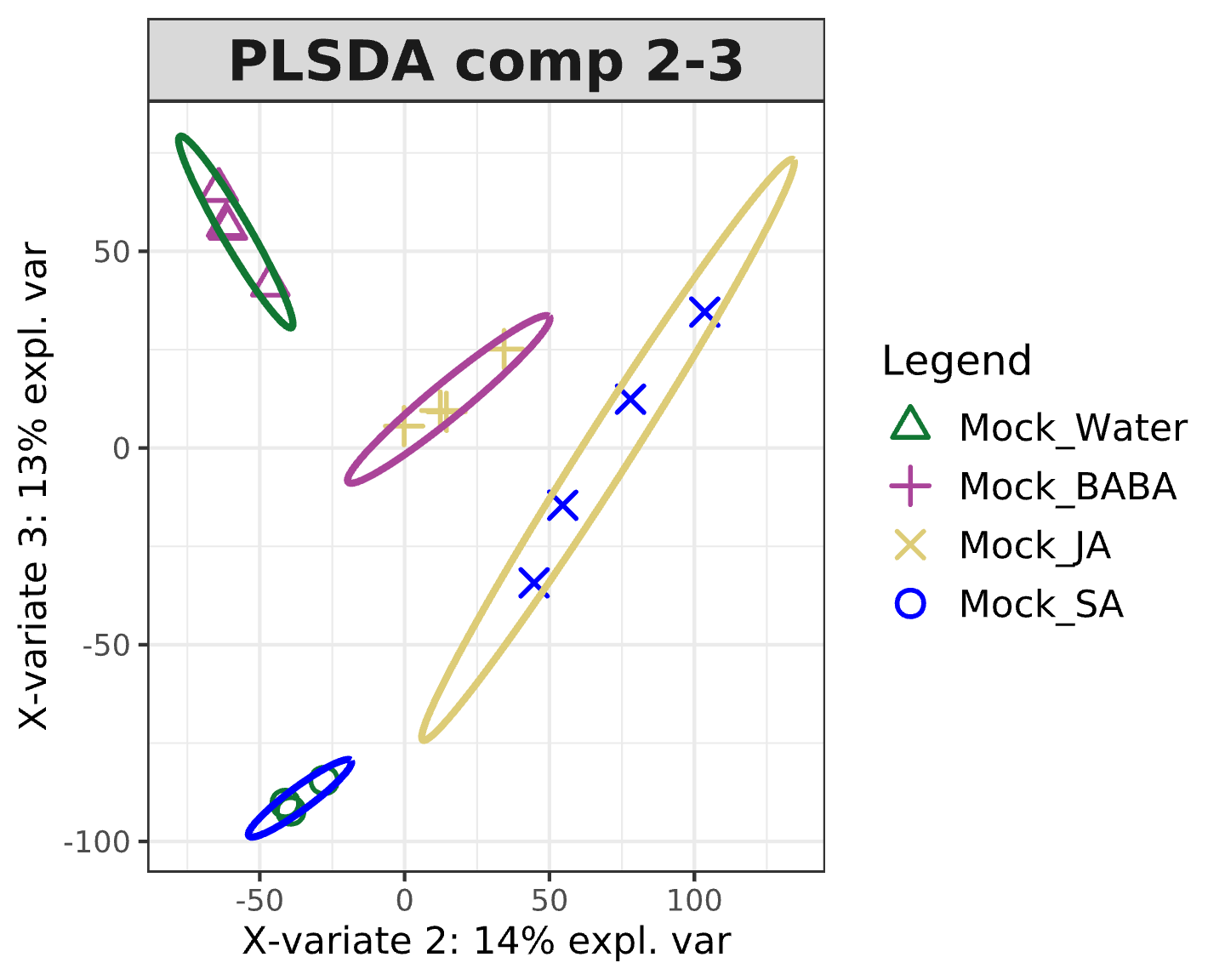
#as far as I can tell from MetaboanalystR page mixOmics is used by Metaboanalyst for PLSDA (sPLS)  
#create initial pls model  
initial\_plsda\_zero\_dpi <- mixOmics::plsda(rlog\_expression\_zero\_dpi, rlog\_classes\_zero\_dpi, ncomp = 4) #have initially set components to 4 then assess performance later  
  
#plot initial pls model for comps 1 and 2  
plotIndiv(initial\_plsda\_zero\_dpi, comp = c(1,2), group = rlog\_classes\_zero\_dpi, ind.names = FALSE,   
 ellipse = TRUE, ellipse.level = 0.95,  
 legend = TRUE, title = 'PLSDA comp 1-2', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))



#plot initial pls model for comps 1 and 3  
plotIndiv(initial\_plsda\_zero\_dpi, comp = c(1,3), group = rlog\_classes\_zero\_dpi, ind.names = FALSE,   
 ellipse = TRUE, ellipse.level = 0.95,  
 legend = TRUE, title = 'PLSDA comp 1-3', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))



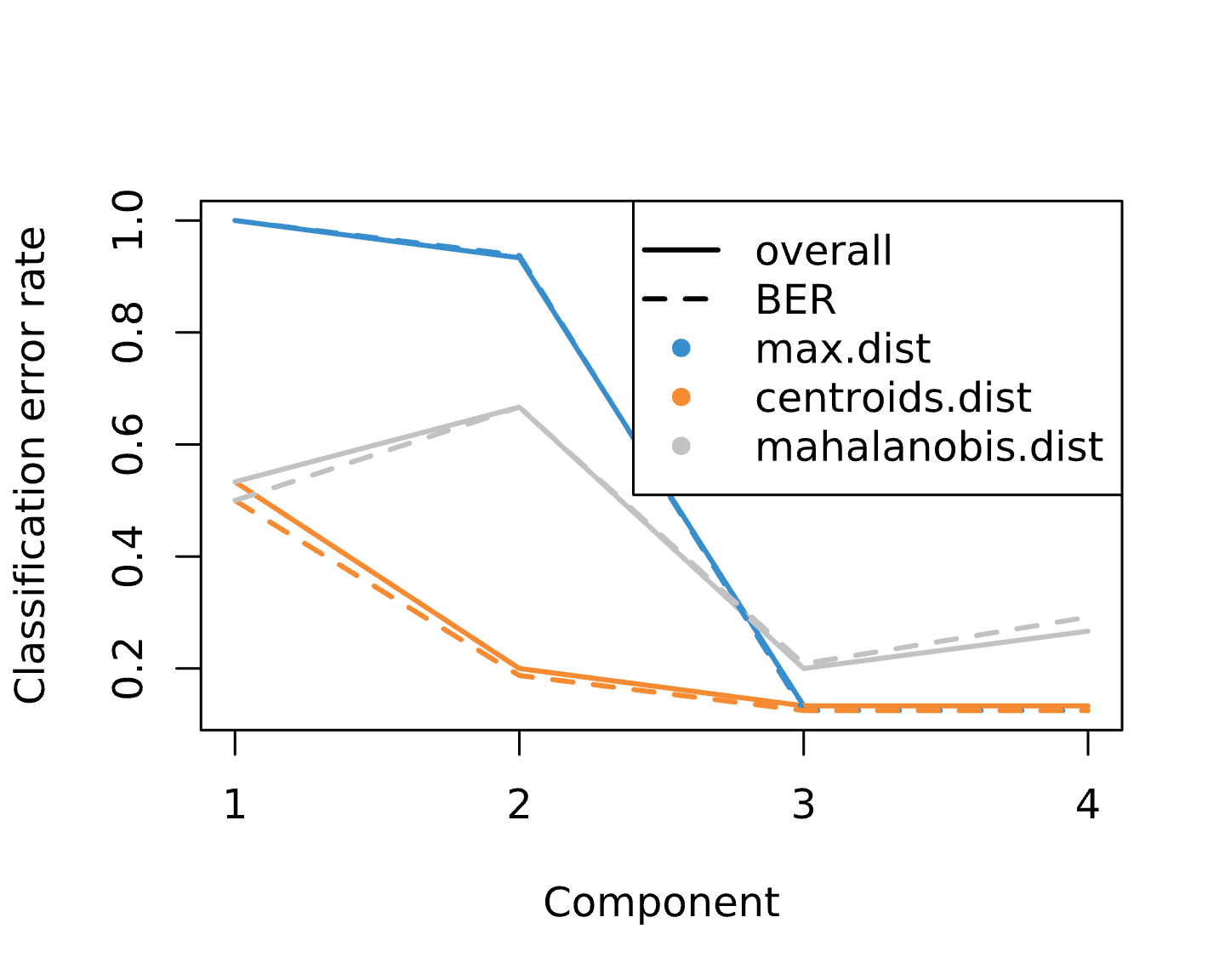
#plot initial pls model for comps 2 and 3  
plotIndiv(initial\_plsda\_zero\_dpi, comp = c(2,3), group = rlog\_classes\_zero\_dpi, ind.names = FALSE,   
 ellipse = TRUE, ellipse.level = 0.95,  
 legend = TRUE, title = 'PLSDA comp 2-3', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))



#note how all three graphs show clearer separation esp 1-2 and 2-3

#PLSDA TUNING

#evaluate initial pls model performance  
perf\_initial\_plsda\_zero\_dpi <- perf(initial\_plsda\_zero\_dpi,   
 validation = "loo", #use leave one out cross-validation  
 auc = TRUE, #calculate area under the curve performance of model  
 dist = "all")   
  
#only works if initial model set up with only 4 components otherwise gives Error in ncomp\_opt[measure, ijk] <- which(t(rowMeans(mat.error.rate[[measure\_i]][[ijk]])) == : number of items to replace is not a multiple of replacement length  
  
plot(perf\_initial\_plsda\_zero\_dpi) #note slight plateau or potential increase in error rate above 3 components



print(perf\_initial\_plsda\_zero\_dpi$choice.ncomp) #this again suggests 3 components

## max.dist centroids.dist mahalanobis.dist  
## overall 3 3 3  
## BER 4 4 3

#note BER is best metric when unequal number of samples per class as here

#FINAL PLSDA SCORES

final\_plsda\_zero\_dpi <- mixOmics::plsda(rlog\_expression\_zero\_dpi, rlog\_classes\_zero\_dpi, ncomp = 3)   
  
#set up pdf to save graphs  
plsda <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_PLSDA\_scores\_plots.pdf"  
pdf(file = plsda)  
  
#plot final pls model for comps 1 and 2  
plotIndiv(final\_plsda\_zero\_dpi, comp = c(1,2), group = rlog\_classes\_zero\_dpi, ind.names = FALSE,   
 ellipse = TRUE, ellipse.level = 0.95, pch = 19, cex = 3,   
 size.xlabel = rel(2), size.ylabel = rel(2), size.axis = rel(1.5),  
 legend = TRUE, title = '0dpi PLSDA: Component 1 vs 2', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))  
  
#plot final pls model for comps 1 and 3  
plotIndiv(final\_plsda\_zero\_dpi, comp = c(1,3), group = rlog\_classes\_zero\_dpi, ind.names = FALSE,   
 ellipse = TRUE, ellipse.level = 0.95, pch = 19, cex = 3,   
 size.xlabel = rel(2), size.ylabel = rel(2), size.axis = rel(1.5),  
 legend = TRUE, title = '0dpi PLSDA: Component 1 vs 3', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))  
  
#plot final pls model for comps 2 and 3  
plotIndiv(final\_plsda\_zero\_dpi, comp = c(2,3), group = rlog\_classes\_zero\_dpi, ind.names = FALSE,   
 ellipse = TRUE, ellipse.level = 0.95, pch = 19, cex = 3,   
 size.xlabel = rel(2), size.ylabel = rel(2), size.axis = rel(1.5),  
 legend = TRUE, title = '0dpi PLSDA: Component 2 vs 3', col.per.group = c("blue", "#117733", "#AA4499", "#DDCC77"))  
  
dev.off()

#FINAL PLSDA LOADINGS

#set up pdf to save graphs  
plsda\_loadings <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_PLSDA\_loadings.pdf"  
pdf(file = plsda\_loadings)  
  
#extract loadings data  
zero\_dpi\_plsda\_loadings <- as.data.frame(final\_plsda\_zero\_dpi$loadings[[1]])  
  
#PC1 vs PC2  
ggplot(data = zero\_dpi\_plsda\_loadings)+  
 geom\_point(mapping = aes(x = comp1, y = comp2))+  
 scale\_y\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 scale\_x\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 xlab("Loadings 1")+  
 ylab("Loadings 2")+  
 ggtitle("0dpi loadings plot")  
  
#PC1 vs PC3  
ggplot(data = zero\_dpi\_plsda\_loadings)+  
 geom\_point(mapping = aes(x = comp1, y = comp3))+  
 scale\_y\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 scale\_x\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 xlab("Loadings 1")+  
 ylab("Loadings 3")+  
 ggtitle("0dpi loadings plot")  
  
#PC2 vs PC3  
ggplot(data = zero\_dpi\_plsda\_loadings)+  
 geom\_point(mapping = aes(x = comp2, y = comp3))+  
 scale\_y\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 scale\_x\_continuous(limits = c(-0.02,0.02), breaks = seq(-0.02,0.02,0.01))+  
 xlab("Loadings 2")+  
 ylab("Loadings 3")+  
 ggtitle("0dpi loadings plot")  
  
dev.off()

#FINAL PLSDA AUROC

#set up pdf to save graphs  
plsda\_auroc <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_PLSDA\_AUROC.pdf"  
pdf(file = plsda\_auroc)  
  
#check performance of final model using aucroc curve  
auc.plsda.zero.dpi\_1 <- auroc(final\_plsda\_zero\_dpi, roc.comp = 1, print = TRUE)

## $Comp1  
## AUC p-value  
## Mock\_Water vs Other(s) 0.3611 0.470500  
## Mock\_BABA vs Other(s) 0.6591 0.360800  
## Mock\_JA vs Other(s) 1.0000 0.004075  
## Mock\_SA vs Other(s) 0.9545 0.009023  
##   
## $Comp2  
## AUC p-value  
## Mock\_Water vs Other(s) 0.6667 0.386500  
## Mock\_BABA vs Other(s) 1.0000 0.004075  
## Mock\_JA vs Other(s) 1.0000 0.004075  
## Mock\_SA vs Other(s) 1.0000 0.004075  
##   
## $Comp3  
## AUC p-value  
## Mock\_Water vs Other(s) 1 0.009375  
## Mock\_BABA vs Other(s) 1 0.004075  
## Mock\_JA vs Other(s) 1 0.004075  
## Mock\_SA vs Other(s) 1 0.004075

auc.plsda.zero.dpi\_1to2 <- auroc(final\_plsda\_zero\_dpi, roc.comp = 2, print = TRUE)

## $Comp1  
## AUC p-value  
## Mock\_Water vs Other(s) 0.3611 0.470500  
## Mock\_BABA vs Other(s) 0.6591 0.360800  
## Mock\_JA vs Other(s) 1.0000 0.004075  
## Mock\_SA vs Other(s) 0.9545 0.009023  
##   
## $Comp2  
## AUC p-value  
## Mock\_Water vs Other(s) 0.6667 0.386500  
## Mock\_BABA vs Other(s) 1.0000 0.004075  
## Mock\_JA vs Other(s) 1.0000 0.004075  
## Mock\_SA vs Other(s) 1.0000 0.004075  
##   
## $Comp3  
## AUC p-value  
## Mock\_Water vs Other(s) 1 0.009375  
## Mock\_BABA vs Other(s) 1 0.004075  
## Mock\_JA vs Other(s) 1 0.004075  
## Mock\_SA vs Other(s) 1 0.004075

auc.plsda.zero.dpi\_1to3 <- auroc(final\_plsda\_zero\_dpi, roc.comp = 3, print = TRUE) # AUROC for all 3 components

## $Comp1  
## AUC p-value  
## Mock\_Water vs Other(s) 0.3611 0.470500  
## Mock\_BABA vs Other(s) 0.6591 0.360800  
## Mock\_JA vs Other(s) 1.0000 0.004075  
## Mock\_SA vs Other(s) 0.9545 0.009023  
##   
## $Comp2  
## AUC p-value  
## Mock\_Water vs Other(s) 0.6667 0.386500  
## Mock\_BABA vs Other(s) 1.0000 0.004075  
## Mock\_JA vs Other(s) 1.0000 0.004075  
## Mock\_SA vs Other(s) 1.0000 0.004075  
##   
## $Comp3  
## AUC p-value  
## Mock\_Water vs Other(s) 1 0.009375  
## Mock\_BABA vs Other(s) 1 0.004075  
## Mock\_JA vs Other(s) 1 0.004075  
## Mock\_SA vs Other(s) 1 0.004075

dev.off()

#DESEQ ANALYSIS

#estimation of size factors, estimation of dispersion, and Negative Binomial GLM fitting and Wald statistics  
zero\_dpi\_DESeq\_out <- DESeq(zero\_dpi\_raw\_gene\_data)

#EXTRACT DEGs 1 fc

factor='condition'  
reference='Mock\_Water'  
treatment='Mock\_BABA'  
DEGs\_0dpi\_BABA <- results(zero\_dpi\_DESeq\_out, contrast = c(factor, treatment, reference))   
  
print(paste("0dpi BABA =", sum(DEGs\_0dpi\_BABA$padj <= 0.05, na.rm = TRUE), "significant DEGs at padj cutoff 0.05"))

## [1] "0dpi BABA = 1774 significant DEGs at padj cutoff 0.05"

factor='condition'  
reference='Mock\_Water'  
treatment='Mock\_SA'  
DEGs\_0dpi\_SA <- results(zero\_dpi\_DESeq\_out, contrast = c(factor, treatment, reference))  
  
print(paste("0dpi SA =", sum(DEGs\_0dpi\_SA$padj <= 0.05, na.rm = TRUE), "significant DEGs at padj cutoff 0.05"))

## [1] "0dpi SA = 2094 significant DEGs at padj cutoff 0.05"

factor='condition'  
reference='Mock\_Water'  
treatment='Mock\_JA'  
DEGs\_0dpi\_JA <- results(zero\_dpi\_DESeq\_out, contrast = c(factor, treatment, reference))  
  
print(paste("0dpi JA =", sum(DEGs\_0dpi\_JA$padj <= 0.05, na.rm = TRUE), "significant DEGs at padj cutoff 0.05"))

## [1] "0dpi JA = 2111 significant DEGs at padj cutoff 0.05"

#for up and downregulation have used log2 threshold of 1 or -1 and padj cutoff of 0.05  
  
#UP DEGs  
UP\_DEGs\_0dpi\_BABA <- subset(DEGs\_0dpi\_BABA, DEGs\_0dpi\_BABA$log2FoldChange > 1 & DEGs\_0dpi\_BABA$padj <= 0.05)   
  
print(paste("BABA UP DEGs =", length(UP\_DEGs\_0dpi\_BABA$log2FoldChange)))

## [1] "BABA UP DEGs = 703"

UP\_DEGs\_0dpi\_SA <- subset(DEGs\_0dpi\_SA, DEGs\_0dpi\_SA$log2FoldChange > 1 & DEGs\_0dpi\_SA$padj <= 0.05)   
  
print(paste("SA UP DEGs =", length(UP\_DEGs\_0dpi\_SA$log2FoldChange)))

## [1] "SA UP DEGs = 1142"

UP\_DEGs\_0dpi\_JA <- subset(DEGs\_0dpi\_JA, DEGs\_0dpi\_JA$log2FoldChange > 1 & DEGs\_0dpi\_JA$padj <= 0.05)   
  
print(paste("JA UP DEGs =", length(UP\_DEGs\_0dpi\_JA$log2FoldChange)))

## [1] "JA UP DEGs = 628"

#DOWN DEGs  
  
DOWN\_DEGs\_0dpi\_BABA <- subset(DEGs\_0dpi\_BABA, DEGs\_0dpi\_BABA$log2FoldChange < -1 & DEGs\_0dpi\_BABA$padj <= 0.05)   
  
print(paste("BABA DOWN DEGs =", length(DOWN\_DEGs\_0dpi\_BABA$log2FoldChange)))

## [1] "BABA DOWN DEGs = 415"

DOWN\_DEGs\_0dpi\_SA <- subset(DEGs\_0dpi\_SA, DEGs\_0dpi\_SA$log2FoldChange < -1 & DEGs\_0dpi\_SA$padj <= 0.05)   
  
print(paste("SA DOWN DEGs =", length(DOWN\_DEGs\_0dpi\_SA$log2FoldChange)))

## [1] "SA DOWN DEGs = 298"

DOWN\_DEGs\_0dpi\_JA <- subset(DEGs\_0dpi\_JA, DEGs\_0dpi\_JA$log2FoldChange < -1 & DEGs\_0dpi\_JA$padj <= 0.05)   
  
print(paste("JA DOWN DEGs =", length(DOWN\_DEGs\_0dpi\_JA$log2FoldChange)))

## [1] "JA DOWN DEGs = 599"

#DATA SUMMARY TABLE

#note run this first, then 1dpi, then 2dpi

#create table  
zero\_dpi\_summary\_data <-  
 data.frame('Timepoint' =   
 c("0 dpi", "0 dpi", "0 dpi"),   
 'Comparison' =   
 c("Mock BABA vs Mock Water", "Mock SA vs Mock Water", "Mock JA vs Mock Water"),  
 'All\_significant\_DEGs' =   
 c(sum(DEGs\_0dpi\_BABA$padj <= 0.05, na.rm = TRUE), sum(DEGs\_0dpi\_SA$padj <= 0.05, na.rm = TRUE), sum(DEGs\_0dpi\_JA$padj <= 0.05, na.rm = TRUE)),  
 'Upregulated\_DEGs' =   
 c(nrow(UP\_DEGs\_0dpi\_BABA), nrow(UP\_DEGs\_0dpi\_SA), nrow(UP\_DEGs\_0dpi\_JA)),  
 'Downregulated\_DEGs' =   
 c(nrow(DOWN\_DEGs\_0dpi\_BABA), nrow(DOWN\_DEGs\_0dpi\_SA), nrow(DOWN\_DEGs\_0dpi\_JA))  
 )  
  
#save as csv  
write.csv(zero\_dpi\_summary\_data, "Deseq\_analysis/zero\_dpi\_out/summary\_data\_0dpi\_fc1.csv", row.names=FALSE)

#VENN DIAGRAMS

#sa = orange, ja = magenta baba = dark-green  
  
#set up pdf to save graphs  
venns <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_Venn\_diagrams\_fc1.pdf"  
pdf(file = venns)  
  
#all DEGs  
sig\_DEGs\_BABA\_0dpi <- subset(DEGs\_0dpi\_BABA, DEGs\_0dpi\_BABA$padj <= 0.05) #significance threshold 0.05  
Venn\_All\_DEGs\_0dpi\_BABA <- row.names(sig\_DEGs\_BABA\_0dpi)  
sig\_DEGs\_SA\_0dpi <- subset(DEGs\_0dpi\_SA, DEGs\_0dpi\_SA$padj <= 0.05)  
Venn\_All\_DEGs\_0dpi\_SA <- row.names(sig\_DEGs\_SA\_0dpi)  
sig\_DEGs\_JA\_0dpi <- subset(DEGs\_0dpi\_JA, DEGs\_0dpi\_JA$padj <= 0.05)  
Venn\_All\_DEGs\_0dpi\_JA <- row.names(sig\_DEGs\_JA\_0dpi)  
  
vennPlot <- venn.diagram(  
 list(Venn\_All\_DEGs\_0dpi\_SA, Venn\_All\_DEGs\_0dpi\_JA, Venn\_All\_DEGs\_0dpi\_BABA), filename = NULL,   
 fill=c("#DDCC77", "#AA4499", "#117733"), print.mode = c("raw", "percent"), alpha=c(0.5,0.5,0.5), cex=2.5,  
 category.names = c(" "," ", " "), main = "All sigificant DEGs 0dpi", main.cex = 2, ext.text = TRUE,   
 ext.percent = 0.1, ext.pos = 3, ext.length = 0.7, ext.dist = -0.02)  
  
grid.newpage()  
  
grid.draw(vennPlot)  
  
#upregulated genes  
Venn\_UP\_DEGs\_0dpi\_BABA <- row.names(UP\_DEGs\_0dpi\_BABA)  
Venn\_UP\_DEGs\_0dpi\_SA <- row.names(UP\_DEGs\_0dpi\_SA)  
Venn\_UP\_DEGs\_0dpi\_JA <- row.names(UP\_DEGs\_0dpi\_JA)  
  
vennPlot <- venn.diagram(  
 list(Venn\_UP\_DEGs\_0dpi\_SA, Venn\_UP\_DEGs\_0dpi\_JA, Venn\_UP\_DEGs\_0dpi\_BABA), filename = NULL,   
 fill=c("#DDCC77", "#AA4499", "#117733"), print.mode = c("raw", "percent"), alpha=c(0.5,0.5,0.5), cex=2.5,  
 category.names = c(" "," ", " "), main = "Significant upregulated genes 0dpi", main.cex = 2, ext.text = TRUE,  
 ext.percent = 0.1, ext.pos = 3, ext.length = 0.7, ext.dist = -0.02)  
  
grid.newpage()  
  
grid.draw(vennPlot)  
  
#downregulated genes  
Venn\_DOWN\_DEGs\_0dpi\_BABA <- row.names(DOWN\_DEGs\_0dpi\_BABA)  
Venn\_DOWN\_DEGs\_0dpi\_SA <- row.names(DOWN\_DEGs\_0dpi\_SA)  
Venn\_DOWN\_DEGs\_0dpi\_JA <- row.names(DOWN\_DEGs\_0dpi\_JA)  
  
vennPlot <- venn.diagram(  
 list(Venn\_DOWN\_DEGs\_0dpi\_SA, Venn\_DOWN\_DEGs\_0dpi\_JA, Venn\_DOWN\_DEGs\_0dpi\_BABA), filename = NULL,   
 fill=c("#DDCC77", "#AA4499", "#117733"), print.mode = c("raw", "percent"), alpha=c(0.5,0.5,0.5), cex=2.5,  
 category.names = c(" "," ", " "), main = "Significant downregulated genes 0dpi", main.cex = 2, ext.text = TRUE,  
 ext.percent = 0.1, ext.pos = 3, ext.length = 0.7, ext.dist = -0.02)  
  
grid.newpage()  
  
grid.draw(vennPlot)  
  
dev.off()

#GENE CLUSTERING HEATMAP

#set up pdf to save graphs  
heatmaps <- "Deseq\_analysis/zero\_dpi\_out/zero\_dpi\_heatmaps.pdf"  
pdf(file = heatmaps)  
  
#select significant DEGs for BABA mock  
significant\_BABA\_DEGs\_heatmap <- DEGs\_0dpi\_BABA$padj <= 0.05   
  
sum(significant\_BABA\_DEGs\_heatmap, na.rm = TRUE) #check number = same as before

## [1] 1774

#select significant DEGs for SA mock  
significant\_SA\_DEGs\_heatmap <- DEGs\_0dpi\_SA$padj <= 0.05   
  
sum(significant\_SA\_DEGs\_heatmap, na.rm = TRUE) #check number = same as before

## [1] 2094

#select significant DEGs for JA mock  
significant\_JA\_DEGs\_heatmap <- DEGs\_0dpi\_JA$padj <= 0.05   
  
sum(significant\_JA\_DEGs\_heatmap, na.rm = TRUE) #check number = same as before

## [1] 2111

#combine significant DEGs into single vector = merges duplicates  
significant\_all\_DEGs\_heatmap <- base::Reduce('|', list(significant\_BABA\_DEGs\_heatmap, significant\_SA\_DEGs\_heatmap, significant\_JA\_DEGs\_heatmap))  
  
sum(significant\_all\_DEGs\_heatmap, na.rm = TRUE) #check number = total in venn diagram so correct

## [1] 4377

#create df  
heatmap\_df <- data.frame(assay(rlog\_zero\_dpi\_raw\_gene\_data)[which(significant\_all\_DEGs\_heatmap),])  
  
#remove rows containing all zeros  
heatmap\_df[heatmap\_df==0] <- NA #convert 0 to NAs  
heatmap\_df <- na.omit(heatmap\_df) #remove NAs  
  
#calculate mean  
m = apply(heatmap\_df, 1, mean, na.rm = T)  
  
#calculate sd  
sd = apply(heatmap\_df, 1, sd, na.rm = T)  
  
#scale  
scaled\_heatmap\_df <- (heatmap\_df - m)/sd  
  
# #note scaling is based on the scale\_rows function within pheatmap which performs the following   
# scale\_rows = function(x){  
# m = apply(x, 1, mean, na.rm = T)  
# s = apply(x, 1, sd, na.rm = T)  
# return((x - m) / s)}  
  
#set up gene clustering dendrogram  
gene\_dend <- as.dendrogram(hclust(dist(scaled\_heatmap\_df)))   
#dist calculates distances using euclidean distances as default   
#hclust performs hierarchical clustering using the complete linkage method by default  
  
gene\_dend #height = 7.438649

## 'dendrogram' with 2 branches and 4377 members total, at height 7.438649

#try different heights to cut tree to find optimal number of clusters to colour code  
gene\_dend1 <- color\_branches(gene\_dend, h=7.4, col = brewer.pal(8,"Dark2"))  
plot(gene\_dend1, main = "cut height 7.4") #4 clusters = probably too few, definitely minimum  
  
#based on plot somewhere around 6.5 looks about right  
gene\_dend2 <- color\_branches(gene\_dend, h=6.5, col = brewer.pal(8,"Dark2"))  
plot(gene\_dend2, main = "cut height 6.5") #12 clusters = too many  
  
gene\_dend3 <- color\_branches(gene\_dend, h=6.6, col = brewer.pal(8,"Dark2"))  
plot(gene\_dend3, main = "cut height 6.6") #11 clusters = too many  
  
gene\_dend4 <- color\_branches(gene\_dend, h=6.7, col = brewer.pal(8,"Dark2"))  
plot(gene\_dend4, main = "cut height 6.7") #10 clusters = possibly too many  
  
gene\_dend5 <- color\_branches(gene\_dend, h=6.8, col = brewer.pal(8,"Dark2"))  
plot(gene\_dend5, main = "cut height 6.8") #10 clusters = possibly too many  
  
gene\_dend6 <- color\_branches(gene\_dend, h=6.9, col = brewer.pal(8,"Dark2"))  
plot(gene\_dend6, main = "cut height 6.9") #8 clusters = looks good  
  
#save optimal number of clusters as k  
cut <- cutree(gene\_dend, h=6.9)  
k <- as.numeric(max(levels(as.factor(cut))))  
  
#set up sample annotations for heatmap  
sample\_annotations <- zero\_dpi\_sample\_info[,3]  
conditions <- HeatmapAnnotation(Condition = sample\_annotations, col = list(Condition = c("Mock\_Water" = "blue", "Mock\_BABA" = "#117733", "Mock\_JA" = "#AA4499", "Mock\_SA" = "#DDCC77")), gp = gpar(col = "black"))  
  
  
#plot heatmap  
htmp1 <- ComplexHeatmap::Heatmap(scaled\_heatmap\_df, cluster\_columns = FALSE,   
 cluster\_rows = gene\_dend6, #use optimal cluster dendrogram for colour coding  
 split = k, #split up dendrogram to number clusters  
 col = viridis(7), show\_row\_names = FALSE, show\_column\_names = FALSE,  
 heatmap\_legend\_param = list(title = NULL, direction = "horizontal"),   
 top\_annotation = conditions)

## `use\_raster` is automatically set to TRUE for a matrix with more than  
## 2000 rows. You can control `use\_raster` argument by explicitly setting  
## TRUE/FALSE to it.  
##   
## Set `ht\_opt$message = FALSE` to turn off this message.

draw(htmp1, heatmap\_legend\_side = "bottom")  
  
dev.off()

#SAVE COUNTS DATA FOR INTEGRATION

#select counts data  
gene\_counts\_0dpi <- as.data.frame(counts(zero\_dpi\_raw\_gene\_data))   
  
#add column of gene names  
gene\_counts\_0dpi <- cbind(gene = row.names(gene\_counts\_0dpi), gene\_counts\_0dpi)  
  
#save counts data  
write.csv(gene\_counts\_0dpi,   
 file = "Deseq\_analysis/zero\_dpi\_out/gene\_counts\_0dpi.csv",  
 row.names = FALSE)

#SAVE DATA FOR GO TERM ANALYSIS

#note run this first, then 1dpi, then 2dpi

save(zero\_dpi\_raw\_gene\_data,   
 DEGs\_0dpi\_BABA,   
 DEGs\_0dpi\_SA,   
 DEGs\_0dpi\_JA,   
 file = "GO\_analysis/R\_topGO/Initial\_GO\_input\_data\_fc1.RData")