ProcessClustering

## Process Clustering Results

Load libraries:

library(pheatmap)  
library(edgeR)

## Loading required package: limma

library(plyr)  
library(ggplot2)  
library(stringr)  
library(reshape2)  
library(pastecs)

## Warning: package 'pastecs' was built under R version 4.0.5

library(igraph)

## Warning: package 'igraph' was built under R version 4.0.5

##   
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':  
##   
## decompose, spectrum

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

library(RColorBrewer)  
library(data.table)

## Warning: package 'data.table' was built under R version 4.0.5

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:pastecs':  
##   
## first, last

## The following objects are masked from 'package:reshape2':  
##   
## dcast, melt

library(png)  
library(knitr)

## Warning: package 'knitr' was built under R version 4.0.5

## Read in table of counts per cluster and get it formatted correctly

mydata <- read.csv(file="GroupStatsPerCluster.csv")  
mydata.orig <- mydata  
mydata <- mydata[,1:3]   
melted <- melt(mydata, id.vars = c("Cluster", "Term"))

## Warning in melt(mydata, id.vars = c("Cluster", "Term")): The melt generic in  
## data.table has been passed a data.frame and will attempt to redirect to the  
## relevant reshape2 method; please note that reshape2 is deprecated, and this  
## redirection is now deprecated as well. To continue using melt methods from  
## reshape2 while both libraries are attached, e.g. melt.list, you can prepend the  
## namespace like reshape2::melt(mydata). In the next version, this warning will  
## become an error.

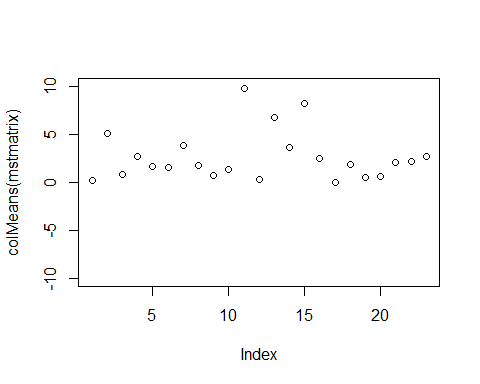
casted <- dcast(melted, Term ~ variable + Cluster)

## Warning in dcast(melted, Term ~ variable + Cluster): The dcast generic  
## in data.table has been passed a data.frame and will attempt to redirect  
## to the reshape2::dcast; please note that reshape2 is deprecated, and this  
## redirection is now deprecated as well. Please do this redirection yourself like  
## reshape2::dcast(melted). In the next version, this warning will become an error.

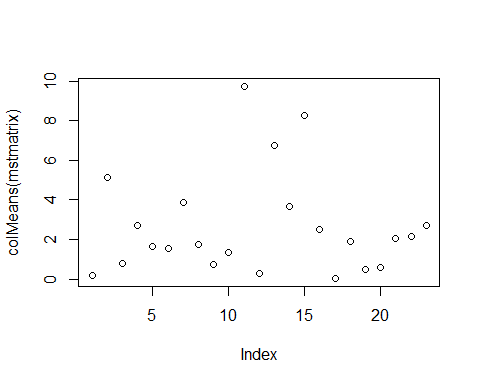
mydata <- casted  
rm(casted)  
  
for(i in c(2:ncol(mydata))) {  
 mydata[,i] <- as.numeric(as.character(mydata[,i]))  
}  
#set row names  
rownames(mydata) <- mydata[,1]   
#remove redundant 1st col  
mydata[,1] <- NULL   
#take out some useless text  
colnames(mydata) <- gsub("Count\_", "", colnames(mydata))   
rownames(mydata) <- gsub("\_processed", "", rownames(mydata))

Spanning tree

#from copy/paste aggregated data from vortex  
mstdata.orig <- read.csv(file = "ClusterFeatureAverages.csv")  
#extract RGB values for cluster colors  
vertexColors <- str\_match\_all(mstdata.orig$Color, "[0-9]{1,3}")   
#convert to hex  
vertexColors <- sapply(vertexColors, function(x)   
 rgb(x[1], x[2], x[3], maxColorValue=255))  
  
#keep only cluster IDs and feature avg values  
mstdata <- mstdata.orig[,-c(1,3,4,5)]   
rownames(mstdata) <- mstdata$ClusterID  
mstdata$ClusterID <- NULL  
  
for(i in c(2:ncol(mstdata))) {  
 mstdata[,i] <- as.numeric(as.character(mstdata[,i]))  
}  
  
mstmatrix <- as.matrix(mstdata)  
#if not clustering on all params, need to scale params not used for clustering  
#all values should be near 0, very near  
plot(colMeans(mstmatrix), ylim = c(-10,10))



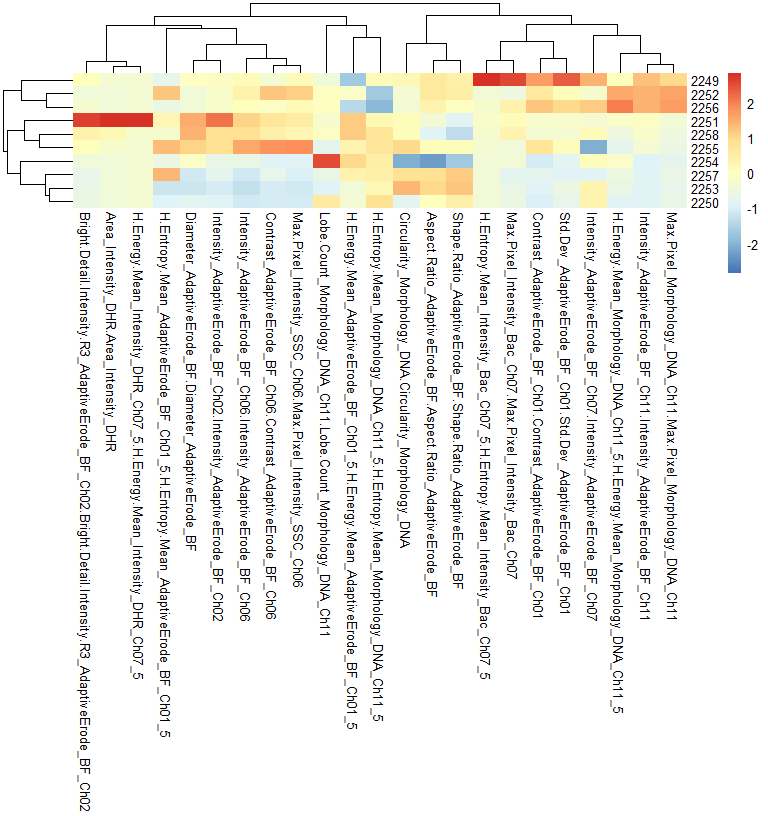
#mstmatrix <- scale(mstmatrix[,c(10:27)])  
plot(colMeans(mstmatrix))



#generate distance matrix object  
mydist <- dist(mstmatrix, method = 'euclidean', diag =TRUE, upper = TRUE)   
#convert to matrix object  
distmat <- as.matrix(mydist)

Plot heatmap of clusters by feature averages, can get sense of over clustering if many clusters have highly similar means across channels

pheatmap(mstdata, scale = "column")



Convert to igraph object and make minimum spanning tree

#create adjacency matrix  
g <- graph.adjacency(distmat, weighted = TRUE)   
mymst <- mst(g)  
  
V(mymst)$color <- vertexColors  
  
list.vertex.attributes(mymst)

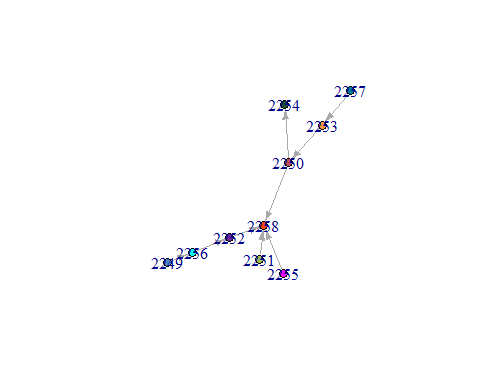
## [1] "name" "color"

Make a layout so X/Y coords are accessible

layout <- layout\_with\_fr(mymst, dim = 2, niter = 1000)  
  
treecoords <- as.data.frame(layout)  
rownames(treecoords) <- rownames(mstdata)  
colnames(treecoords) <- c("MST-X","MST-Y")  
#save this and integrate to big csv of all file events using "parse big csv" code  
write.csv(treecoords, file = "MSTcoords.csv")

Plot MST, all nodes same size, color by cluster ID to match FDL plots

plot(mymst, layout = layout, vertex.size = 10, edge.arrow.size = 0.5)



## Parse Big Master Csv

Accepts as input “ClusterIDs.csv”, “FDL\_coords.csv” and “MSTcoords.csv”. Generates “AllData.csv” for making FDL plots in R and individual csv files for FCS Express R import

#read in csv file exported from Vortex  
mydata1 <- fread("ClusterIDs.csv")   
#read in force directed layout coordinates  
FDLdata <- fread("FDL\_coords.csv", sep = ";")   
colnames(FDLdata) <- c("EventID","Filename","Index\_In\_File","FDL-X","FDL-Y")  
MSTdata <- fread("MSTcoords.csv")  
colnames(MSTdata) <- c("ClusterID","MST-X","MST-Y")  
MSTdata$ClusterID <- as.numeric(as.character(MSTdata$ClusterID))  
FDLdata$EventID <- as.numeric(as.character(FDLdata$EventID))  
#pad eventID for sorting   
mydata1$EventID <- sprintf("%07d", mydata1$EventID)   
#pad these too   
FDLdata$EventID <- sprintf("%07d", FDLdata$EventID)   
  
#order rows by eventID  
mydata1 <- mydata1[order(mydata1$EventID),]   
#order rows by eventID  
FDLdata <- FDLdata[order(FDLdata$EventID),]   
  
#remove duplicate cols with mydata1  
FDLdata[,c(2,3)] <- NULL   
  
#merge using data.table, fast!  
mydata2 <- merge(mydata1, FDLdata, by = "EventID", all.x=TRUE)   
mydata2 <- merge(mydata2, MSTdata, by = "ClusterID", all.x=TRUE)  
  
mydata3 <- na.omit(mydata2, cols = "FDL-Y")  
  
#to use for coloring graphml FDL plot by clusters  
fwrite(mydata3, file = "AllData.csv")   
  
#split big list into list of dataframes based on file name of orig files  
X <- split(mydata2, mydata2$`File Name`)   
  
#sort by original event numbers  
X <- lapply(X, function(x) x[order(x$'Index in File'),])

Save a csv for each data frame in the list of frames

lapply(1:length(X), function(i) write.csv(X[[i]],   
 file = paste0(names(X[i]), ".csv"),  
 row.names = FALSE))

## [[1]]  
## NULL  
##   
## [[2]]  
## NULL  
##   
## [[3]]  
## NULL  
##   
## [[4]]  
## NULL  
##   
## [[5]]  
## NULL  
##   
## [[6]]  
## NULL  
##   
## [[7]]  
## NULL  
##   
## [[8]]  
## NULL  
##   
## [[9]]  
## NULL

## Force Directed Layout Graphs using graphml object

Takes graphml file output from vortex as input, colors using info from tabular data in “AllData.csv”

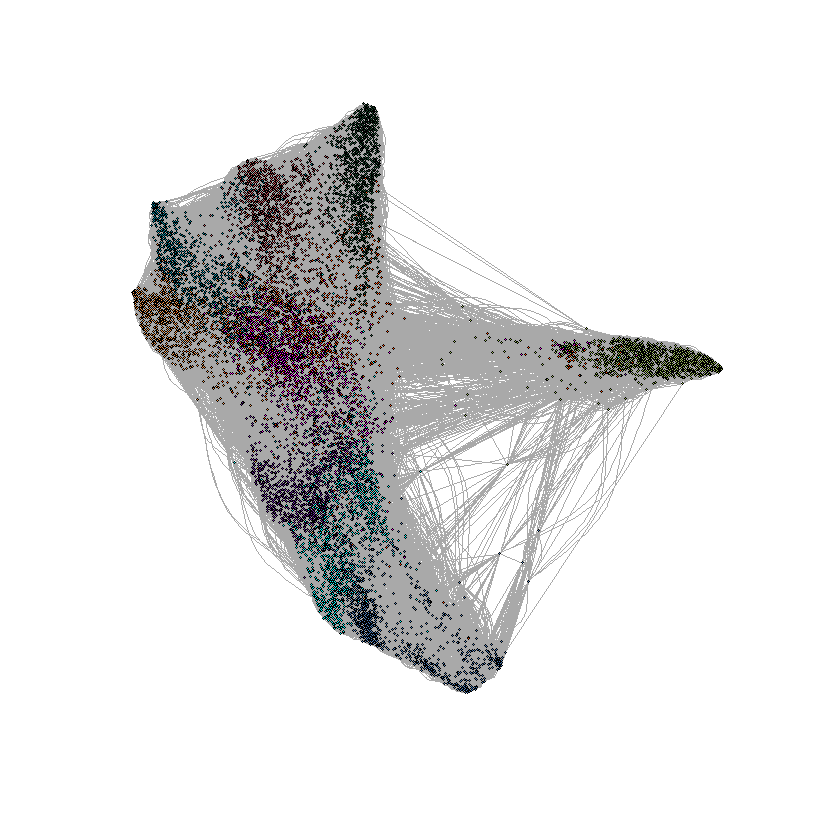
#Open this, comes from "ParseBigCsv" script  
FDL.all <- fread("AllData.csv")   
  
FDL.orig <- FDL.all  
  
#just take these cols out  
FDL.all <- FDL.all[,c("ClusterID","EventID","File Name","Index in File",  
 "FDL-X","FDL-Y")]   
  
#uses for key of clusters to colors  
colorkey <- as.data.frame(V(mymst)$name)   
colorkey$Color <- V(mymst)$color  
  
colnames(colorkey) <- c("ClusterID","Color")  
  
#convert to data table for fast merge later  
colorkey <- data.table(colorkey)   
  
colorkey$ClusterID <- factor(as.character(colorkey$ClusterID))  
  
FDL <- read\_graph(file = "FDL.graphml", format = "graphml")  
  
#shows all vertex attributes  
list.vertex.attributes(FDL)

## [1] "label" "r" "g" "b"   
## [5] "x" "y" "size" "selected"   
## [9] "clusterNode" "groupID" "expValue" "cluster"   
## [13] "clusterZeroBased" "dpID" "id"

#pull out vertex list of cluster names for merging in color info  
gmlclusters <- vertex\_attr(FDL, "cluster")   
  
gmlclusters <- data.table(gmlclusters)  
  
colnames(gmlclusters) <- "ClusterID"  
  
#factor cluster IDs  
gmlclusters$ClusterID <- factor(as.character(gmlclusters$ClusterID))   
#add column of numbers to use for sorting later  
gmlclusters$Sort <- rownames(gmlclusters)   
#transform in a numerical vector  
gmlclusters$Sort <- as.numeric(as.character(gmlclusters$Sort))   
#pad zeros  
gmlclusters$Sort <- sprintf("%05d",gmlclusters$Sort)   
  
#combine colorkey and gmclusters table  
gmlclusters1 <- merge(gmlclusters, colorkey, by = "ClusterID", all.x = TRUE)   
  
#reorder using sort column of numbers  
gmlclusters1 <- gmlclusters1[order(Sort),]   
  
#add color info to igraph object  
V(FDL)$color <- gmlclusters1$Color   
  
#flip Y axis values to match vortex plots  
V(FDL)$y <- -1 \* V(FDL)$y

PLot the force directed graph

plot(FDL, vertex.size = 1)

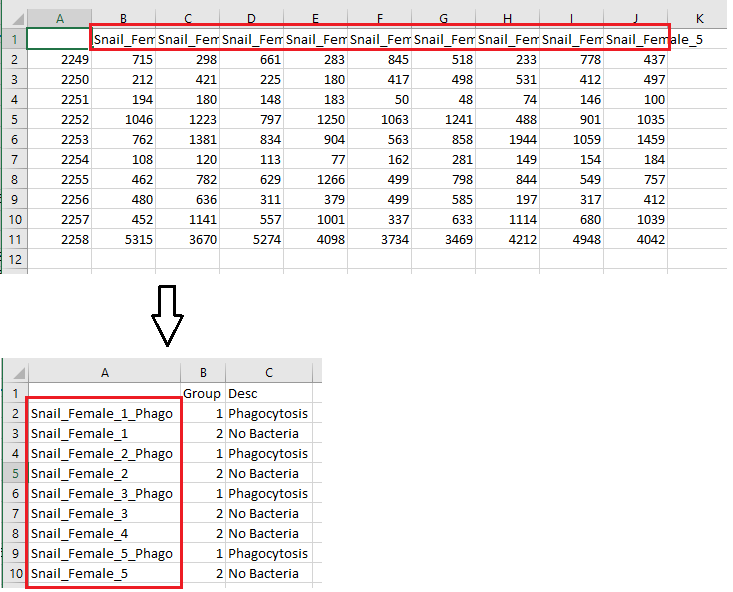


## Use dataframe from above section as counts table, doing this part now to make groupKey for coloring FDL plots by condition

Export the raw counts file, to use for creating an annotation data.frame

#transpose it first  
counts <- t(mydata)   
#deal with this in excel to make group list  
write.csv(counts, file="counts.csv")

Manipulate the “counts.csv” file in Excel as follows to create a file in the proper shape for annotations. Or, use R console if desired.

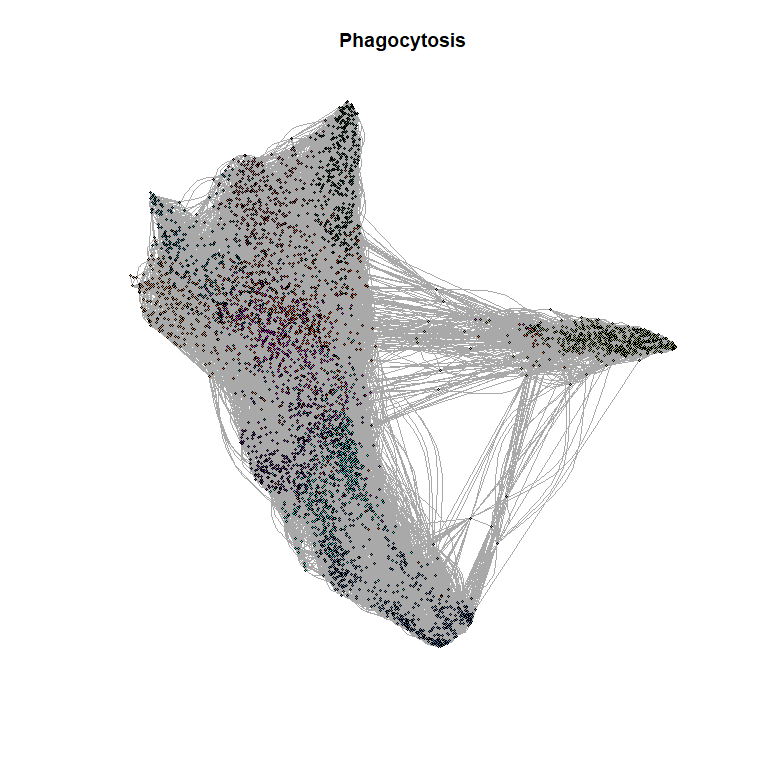
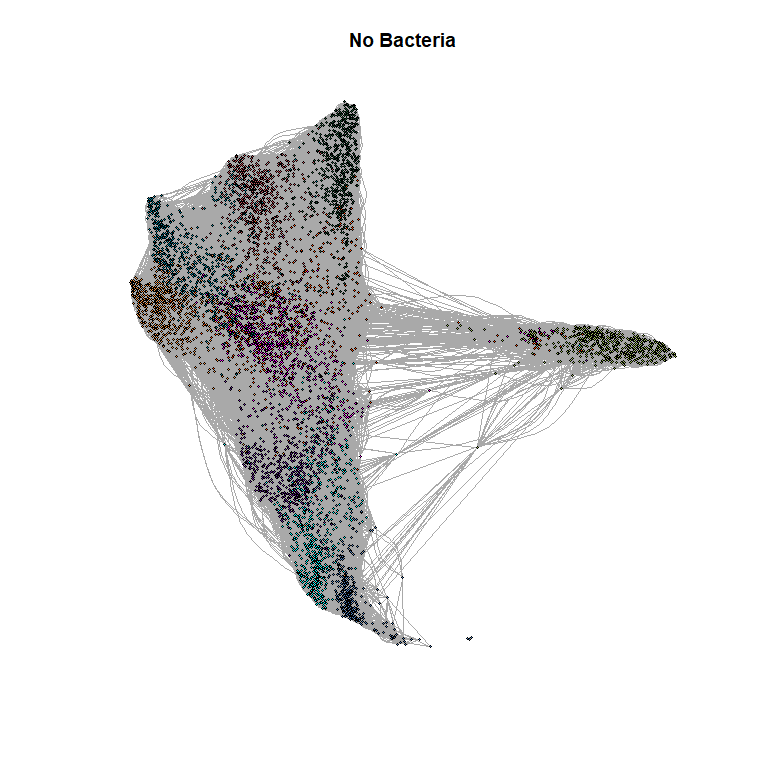


#read it back in, make 1st col group numbers  
groupKey <- read.csv(file = "RowLabels.csv")   
#keep only groupKey rows to match files used  
groupKey <- groupKey[groupKey$X %in% rownames(mydata),]   
rownames(groupKey) <- groupKey$X   
groupKey$X <- NULL  
#refactor  
groupKey$Desc <- as.character(groupKey$Desc)   
groupKey$Desc <- factor(groupKey$Desc)   
#factor groups  
groupKey$Group <- factor(groupKey$Group)   
#levels(groupKey$Group) <- c("1","2","3","4","5","6") #renumber for convenience  
write.csv(groupKey, file = "groupKey.csv")

Setup key to color by cond

#now work with this b/c has sample info that can be parsed to treatment/condition  
FDL.all <- FDL.all[order(EventID),]   
  
colnames(FDL.all) <- c("ClusterID","EventID","FileName","IndexInFile","FDL-X","FDL-Y")  
FDL.all$FileName <- gsub("\_processed", "", FDL.all$FileName)  
  
#make a merge key data frame  
temp1 <- groupKey   
temp1$FileName <- rownames(temp1)   
#remove column with group numbers  
temp1[,1] <- NULL   
#number row names  
row.names(temp1) <- 1:nrow(temp1)   
temp1 <- data.table(temp1)  
   
#seems to be event number, data point ID?  
gmlclusters2 <- as.data.frame(vertex\_attr(FDL, "dpID"))   
gmlclusters2$sort <- rownames(gmlclusters2)  
#transform in a numerical vector  
gmlclusters2$sort <- as.numeric(as.character(gmlclusters2$sort))   
#pad zeros  
gmlclusters2$sort <- sprintf("%05d",gmlclusters2$sort)   
colnames(gmlclusters2) <- c("EventID","Sort")  
gmlclusters2 <- data.table(gmlclusters2)  
   
gmlclusters3 <- merge(FDL.all, gmlclusters2, by = "EventID", all.x = TRUE) #merge the two things  
gmlclusters4 <- merge(gmlclusters3, temp1, by = "FileName")  
  
#reorder using sort column of numbers  
gmlclusters4 <- gmlclusters4[order(Sort),]   
  
  
#add color info to igraph object  
V(FDL)$Treatment <- as.character(gmlclusters4$Desc)   
  
tmts <- unlist(table(V(FDL)$Treatment))  
  
tmts1 <- c(tmts)  
  
tmts <- names(tmts1)  
  
rm(tmts1)  
  
#make subgraph of only specified conditions  
FDL\_AmpFresh <- induced.subgraph(FDL, which(V(FDL)$Treatment=="AmpFresh"))   
  
makeGraphs1 <- function(condName, agraph){  
 x <- eval(condName)  
 x <- induced.subgraph(agraph, which(V(agraph)$Treatment==condName))  
 return(x)  
}  
  
graphSet1 <- lapply(tmts, FUN = makeGraphs1, FDL)  
  
myPlotGraphs <- function(agraph){  
 plot(agraph, vertex.size = 1, main = eval(levels(factor(V(agraph)$Treatment))))  
}

lapply(graphSet1, FUN = myPlotGraphs)



## [[1]]  
## NULL  
##   
## [[2]]  
## NULL

## Use edgeR glm approach for statistical analysis, employs a negative binomial distribution model

#define list of sample groups #make group list for modeling  
group <- groupKey[,'Group']   
  
## make DGElist  
mylist <- DGEList(counts = counts, group = group)  
write.csv(mylist$samples, file="AnnotatedSamples.csv")  
dim(mylist)

## [1] 10 9

mylist.full <- mylist  
summary(cpm(mylist))

## Snail\_Female\_1\_Phago Snail\_Female\_1 Snail\_Female\_2\_Phago Snail\_Female\_2   
## Min. : 11082 Min. : 12180 Min. : 11834 Min. : 8003   
## 1st Qu.: 27909 1st Qu.: 33369 1st Qu.: 25814 1st Qu.: 21619   
## Median : 48328 Median : 71965 Median : 62101 Median : 66677   
## Mean :100000 Mean :100000 Mean :100000 Mean :100000   
## 3rd Qu.: 76980 3rd Qu.:122056 3rd Qu.: 79904 3rd Qu.:123454   
## Max. :545352 Max. :372513 Max. :552309 Max. :425943   
## Snail\_Female\_3\_Phago Snail\_Female\_3 Snail\_Female\_4 Snail\_Female\_5\_Phago  
## Min. : 6121 Min. : 5376 Min. : 7562 Min. : 14682   
## 1st Qu.: 43702 1st Qu.: 56333 1st Qu.: 21051 1st Qu.: 34267   
## Median : 61085 Median : 68205 Median : 52064 Median : 61796   
## Mean :100000 Mean :100000 Mean :100000 Mean :100000   
## 3rd Qu.: 94810 3rd Qu.: 94412 3rd Qu.:106939 3rd Qu.: 87515   
## Max. :457094 Max. :388509 Max. :430411 Max. :497587   
## Snail\_Female\_5   
## Min. : 10038   
## 1st Qu.: 41985   
## Median : 62939   
## Mean :100000   
## 3rd Qu.:104196   
## Max. :405742

#reset libary sizes  
mylist$samples$lib.size <- colSums(mylist$counts)  
mylist$samples

## group lib.size norm.factors  
## Snail\_Female\_1\_Phago 1 9746 1  
## Snail\_Female\_1 2 9852 1  
## Snail\_Female\_2\_Phago 1 9549 1  
## Snail\_Female\_2 2 9621 1  
## Snail\_Female\_3\_Phago 1 8169 1  
## Snail\_Female\_3 2 8929 1  
## Snail\_Female\_4 2 9786 1  
## Snail\_Female\_5\_Phago 1 9944 1  
## Snail\_Female\_5 2 9962 1

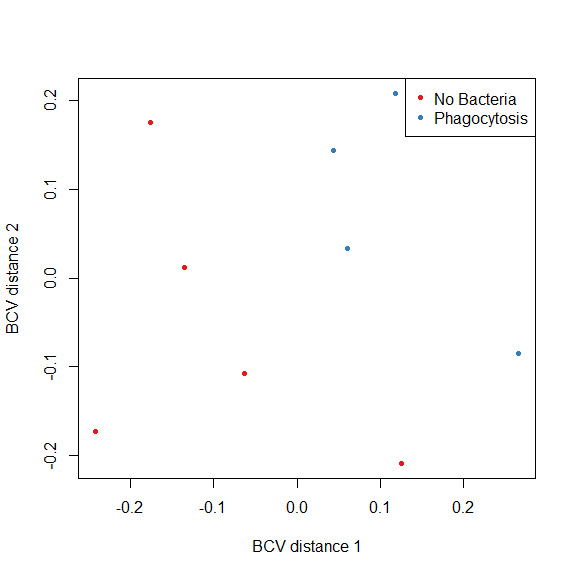
#Normalize  
mylist <- calcNormFactors(mylist, method = "none")  
mylist

## An object of class "DGEList"  
## $counts  
## Snail\_Female\_1\_Phago Snail\_Female\_1 Snail\_Female\_2\_Phago Snail\_Female\_2  
## 2249 715 298 661 283  
## 2250 212 421 225 180  
## 2251 194 180 148 183  
## 2252 1046 1223 797 1250  
## 2253 762 1381 834 904  
## 2254 108 120 113 77  
## 2255 462 782 629 1266  
## 2256 480 636 311 379  
## 2257 452 1141 557 1001  
## 2258 5315 3670 5274 4098  
## Snail\_Female\_3\_Phago Snail\_Female\_3 Snail\_Female\_4 Snail\_Female\_5\_Phago  
## 2249 845 518 233 778  
## 2250 417 498 531 412  
## 2251 50 48 74 146  
## 2252 1063 1241 488 901  
## 2253 563 858 1944 1059  
## 2254 162 281 149 154  
## 2255 499 798 844 549  
## 2256 499 585 197 317  
## 2257 337 633 1114 680  
## 2258 3734 3469 4212 4948  
## Snail\_Female\_5  
## 2249 437  
## 2250 497  
## 2251 100  
## 2252 1035  
## 2253 1459  
## 2254 184  
## 2255 757  
## 2256 412  
## 2257 1039  
## 2258 4042  
##   
## $samples  
## group lib.size norm.factors  
## Snail\_Female\_1\_Phago 1 9746 1  
## Snail\_Female\_1 2 9852 1  
## Snail\_Female\_2\_Phago 1 9549 1  
## Snail\_Female\_2 2 9621 1  
## Snail\_Female\_3\_Phago 1 8169 1  
## Snail\_Female\_3 2 8929 1  
## Snail\_Female\_4 2 9786 1  
## Snail\_Female\_5\_Phago 1 9944 1  
## Snail\_Female\_5 2 9962 1

***must specify # of analysis groups for the “numGrps” variable below***

Maximum 9 unless you change from “Set1” to another such as “Set3”

numGrps <- 2  
mycolors <- brewer.pal(9, "Set1")  
jColors <- with(groupKey,  
 data.frame(Desc = levels(Desc),  
 color = mycolors[1:numGrps]))  
  
  
plotMDS(mylist, method="bcv",  
 col = jColors$color[match(groupKey$Desc, jColors$Desc)],  
 pch=20)  
legend("topright", levels(groupKey$Desc),  
 col=jColors$color, pch=20)

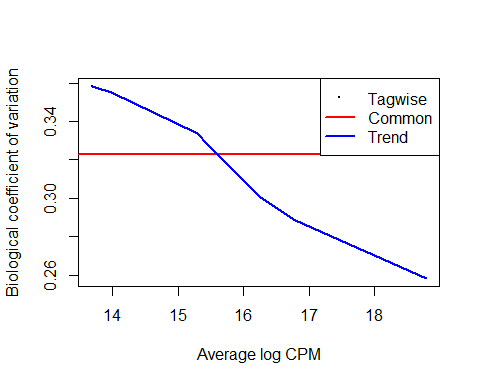


## glm method

#define exp design  
design1 <- model.matrix(~ 0 + group, data=mylist$samples)   
#esimate common dispersion  
y <- estimateDisp(mylist, design1)

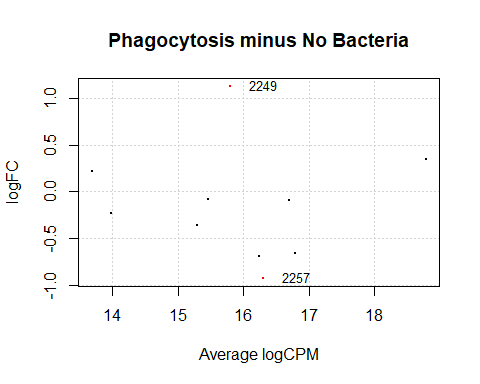
Plot Biological CVs

plotBCV(y)



Fit the model and run a contrast between groups to obtain statistics output

#groups EDTA control vs Phagocytosis   
fit <- glmFit(y, design1) #fit model using design matrix  
  
lrt.one <- glmLRT(fit, contrast = c(1,-1))  
DEG1 <- topTags(lrt.one, n=20, p.value = 0.05)  
write.csv(DEG1, file = "PhagoMinusIce.csv")  
  
plotSmear(lrt.one, de.tags = rownames(DEG1$table), main = "Phagocytosis minus No Bacteria")  
text(x = DEG1$table$logCPM,  
 y = DEG1$table$logFC,  
 labels=rownames(DEG1$table),  
 cex = 0.8,  
 pos = 4,  
 offset = 1)



Below you can see an example of contrasting when there are 4 groups, this chunk is not running in this document however as the example files are from the same group and non clusters can be pulled out by this method as files are too similar.

fit <- glmFit(y, design1) #fit model using design matrix  
#groups Phago Minus Ice  
lrt.one <- glmLRT(fit, contrast = c(-1,1,0,1))   
DEG1 <- topTags(lrt.one, n=20, p.value = 0.05)  
  
#groups Phago Minus EDTA  
lrt.two <- glmLRT(fit, contrast = c(-1,0,0,1))   
DEG2 <- topTags(lrt.two, n=20, p.value = 0.05)  
  
#groups Phago Minus NoBact  
lrt.three <- glmLRT(fit, contrast = c(0,0,-1,1))   
DEG3 <- topTags(lrt.three, n=20, p.value = 0.05)  
  
write.csv(DEG1, file = "PhagoMinusIce.csv")  
write.csv(DEG2, file = "PhagoMinusEDTA.csv")  
write.csv(DEG3, file = "PhagoMinusNoBact.csv")  
  
plotSmear(lrt.one, de.tags = rownames(DEG1$table), main = "Phago Minus Ice")  
text(x = DEG1$table$logCPM,  
 y = DEG1$table$logFC,  
 labels=rownames(DEG1$table),  
 cex = 0.8,  
 pos = 4,  
 offset = 1)  
  
plotSmear(lrt.two, de.tags = rownames(DEG2$table), main = "Phago Minus EDTA")  
text(x = DEG2$table$logCPM,  
 y = DEG2$table$logFC,  
 labels=rownames(DEG2$table),  
 cex = 0.8,  
 pos = 4,  
 offset = 1)  
  
plotSmear(lrt.three, de.tags = rownames(DEG3$table), main = "Phago Minus NoBact")  
text(x = DEG3$table$logCPM,  
 y = DEG3$table$logFC + 6,  
 labels=rownames(DEG3$table),  
 cex = 0.8,  
 pos = 4,  
 offset = 1)  
  
#Also want to do correlation and throw out outliers  
rowAnnot <- as.data.frame(groupKey$Desc)  
rownames(rowAnnot) <- rownames(mydata)  
#rowAnnot$Group <- NULL  
  
pdf(file = "HeatmapSamplesByClusterCounts.pdf", width = 8, height = 6, onefile = FALSE)  
pheatmap(log(mydata+1), scale = "column", annotation\_row = rowAnnot)  
dev.off()  
graphics.off()  
  
pdf(file = "CorrelationSamplesByClusterCounts.pdf", width = 10, height = 10, onefile = FALSE)  
pheatmap(cor(counts), annotation\_row = rowAnnot)  
dev.off()  
graphics.off()

## Plot MST with vertices sized proportional to counts per cluster

ClusterCounts <- as.data.frame(colSums(mydata))  
colnames(ClusterCounts) <- "TotalCounts"  
ClusterCounts$ClusterID <- rownames(ClusterCounts)  
rownames(ClusterCounts) <- 1:nrow(ClusterCounts)  
vertexatt <- merge(ClusterCounts, colorkey, by = "ClusterID")  
vertexatt1 <- as.data.frame(V(mymst)$name)  
colnames(vertexatt1) <- "ClusterID"  
vertexatt1$Sort <- rownames(vertexatt1)  
#as numeric  
vertexatt1$Sort <- as.numeric(as.character(vertexatt1$Sort))   
#pad zeros  
vertexatt1$Sort <- sprintf("%05d",vertexatt1$Sort)   
vertexatt1 <- data.table(vertexatt1)  
vertexatt <- data.table(vertexatt)  
vertexatt <- merge(vertexatt, vertexatt1, by = "ClusterID")  
vertexatt <- vertexatt[order(Sort),]  
  
#calculate CDF for cluster counts distribution  
scaledvert <- ecdf(vertexatt$TotalCounts)   
  
#apply to cluster counts and scale to size nodes of graph  
V(mymst)$size <- scaledvert(vertexatt$TotalCounts) \* 10   
  
#put into dataframe just to have it  
vertexatt$CumDist <- scaledvert(vertexatt$TotalCounts)   
  
#plot new MST, node sizes proportional to counts, nodes color to match FDL plots  
  
#specifying layout here determines coordinates  
plot(mymst, layout = layout,  
 edge.arrow.size = 0.5,  
 label.dist = 1,  
 vertex.color = vertexColors)

