LabBook_27_05_16

Claire Green

Monday

Finalised methods for DEG analysis. Code for microarray:

```
##Differential Expression of Genes##

setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/GeneExpressionAnalysis/Microarray/VCP/")
library(widgetTools)
library(tcltk)
library(DynDoc)
library(tools)
library(affy)
library(Biobase)
library(tkWidgets)

#run program to choose .CEL files from directory
celfiles <- fileBrowser(textToShow = "Choose CEL files", testFun = hasSuffix("[cC][eE][lL]"))
#celfiles<-basename(celfiles)
Data<-ReadAffy(filenames=celfiles) #read in files
rmaEset<-rma(Data) #normalise using RMA
analysis.name<-"VCP" #Label analysis
dataMatrixAll<-exprs(rmaEset) #takes expression from normalised expression set


#mas5call generates presence/absence calls for each probeset
mas5call<-mas5calls(Data)
callMatrixAll<-exprs(mas5call)
colnames(callMatrixAll)<-sub(".CEL", ".mas5-Detection", colnames(callMatrixAll),fixed=TRUE)
colnames(callMatrixAll)<-sub(".cel", ".mas5-Detection", colnames(callMatrixAll),fixed=TRUE)
callMatrixAll<-as.data.frame(callMatrixAll)
callMatrixAll$ProbeSetID<-rownames(callMatrixAll)
countPf<-function(x){
  sum(x=="P")
}

#count how many samples have presence calls
countPl<-apply(callMatrixAll, 1, countPf)
callMatrixAll$ProbeSetID<-rownames(callMatrixAll)
countPdf<-data.frame(ProbeSetID=names(countPl), countP=countPl)

#read annotation file

###USING BIOMART
# library (biomaRt)
# mart <- useMart("ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl", host="www.ensembl.org")
# x <- rownames(dataMatrixAll) #create vector containing probe IDs
# mart_attribute <- listAttributes(mart)
# annotation <- getBM(attributes=c("affy_hg_u133a_2", "hgnc_symbol", "description"),
```

```
#                     filters = "affy_hg_u133a_2", values = x, mart = mart)
# annotation<-subset(annotation, subset=(hgnc_symbol !="")) #if no gene symbol, discount

# USING ANNOTATION FILE (if .csv, convert to .txt using excel)
annotation.file<-"/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/HG-U133_Plus_2.na35.annot.csv/HG-U
#annotation.file<-"/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/HG-U133A_2.na35.annot.csv/HG-U13
#annotation<-read.table(annotation.file, header = TRUE, row.names=NULL, sep="\t", skip=0, stringsAsFacto
dim(annotation)
nrow(annotation)
#[1] 39699
annotation<-subset( annotation, subset=(Gene.Symbol !="---")) #if no gene symbol, discount

# Remove rows in which genes are noted to have negative strand matching probes
idxNegativeStrand<-grep("Negative Strand Matching Probes", annotation$Annotation.Notes)
if(length(idxNegativeStrand)>0)
{
  annotation<-annotation[-idxNegativeStrand,]
}


expressionMatrix<-exprs(rmaEset)
colnames(expressionMatrix)

#this is for matched samples
#tonsil<-factor(c("T101","T101","T102","T102","T103","T103"))
Treat<-factor(rep(c("Control", "Patient"),c(3,7)), levels=c("Control", "Patient"))
design<-model.matrix(~Treat)
rownames(design)<-colnames(expressionMatrix)
design

#Conduct statistical analysis of expression
library(limma)
fit<-lmFit(expressionMatrix, design) #linear model fit
fit<-eBayes(fit)
result<-topTable(fit, coef="TreatPatient", adjust="BH", number=nrow(expressionMatrix)) #"BH" adjust for
#toptable normally takes top number but this takes all


result$"ProbeSetID"<-rownames(result) #make probeset IDs the row names
head(result$"ProbeSetID")
result$"Fold Change"<-2^result$logFC
result$"Fold Change"[result$"Fold Change"<1]<-(-1)/result$"Fold Change"[result$"Fold Change"<1] #conver
expressionLinear<-as.data.frame(2^expressionMatrix)
expressionLinear$ProbeSetID<-rownames(expressionLinear)
result<-merge(result, expressionLinear, by.x="ProbeSetID", by.y="ProbeSetID") #merge values into one ar
result<-merge(annotation, result, by.x="Probe.Set.ID", by.y="ProbeSetID")
result<-merge(result, countPdf, by.x="Probe.Set.ID", by.y="ProbeSetID")

setwd(dir = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
write.csv(result, file=paste(analysis.name, "result.csv", sep=""), sep="\t", row.names=FALSE, quote = F

result<-subset(result, Gene.Symbol!="") #removes any probes for which there are no gene symbols
result<-subset(result, subset=(countP>2)) #only takes results that have at least 2 samples with a prese
```

```
# nrow(result)
# foldchange<-1.5
# pvalue<-0.05
# #adj_P_Val<-0.05
# siggenes<-subset(result, subset=(P.Value < pvalue) & abs(logFC) > log2(foldchange))
# #siggenes<-subset(result, subset=(adj.P.Val < 0.05))
# nrow(siggenes)
# siggenesup<-subset(siggenes, subset= logFC > 0)
# siggenesdown<-subset(siggenes, subset=logFC < 0)
# colnames(siggenesup)
# nrow(siggenesup)
# nrow(siggenesdown)
# UpandDown<-intersect(siggenesup$"Gene.Symbol", siggenesdown$"Gene.Symbol")
# length(UpandDown)
#
# UporDown<-subset(siggenes, subset=(!siggenes$"Gene.Symbol"%in% UpandDown))
# upsiggenes<-subset(siggenesup, subset=(!siggenesup$"Gene.Symbol"%in% UpandDown))
# downsiggenes<-subset(siggenesdown, subset=(!siggenesdown$"Gene.Symbol"%in% UpandDown))
# length(unique(siggenes$"Gene.Symbol"))
# uniquesiggenes <- unique(siggenes$Gene.Symbol)
# length(unique(upsiggenes$"Gene.Symbol"))
# length(unique(downsiggenes$"Gene.Symbol"))
# length(unique(UporDown$"Gene.Symbol"))


###Write results to CSV files for consensus analysis
setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
#dir.create(paste("TopGenes", Sys.Date(), sep = "_")) #create directory using the day's date
#Take results, remove duplicate rows for genes, order by adjusted p value and take top X number of gene
uniqueresult <- result[!duplicated(result[,15]),]

#For ordering by adjusted p value
genesort <- uniqueresult[order(uniqueresult$adj.P.Val),]
write.csv(genesort, file=paste(analysis.name, "rankeduniqueresult.csv", sep=""), sep="\t", row.names=FA
```

Code for Ravits:

```
##RNA-Seq Gene Expression Analysis using Limma##

analysis.name<-"RAV" #Label analysis
setwd(dir = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/GeneExpressionAnalysis/RNA-seq/Ravits/
# Counts <- read.table(file = 'GSE67196_Petrucelli2015_ALS_genes.rawcount.txt', header = TRUE)
#
# write.csv(x = Counts, file = "counts_petrucelli.csv")

Counts <- read.csv(file = "ravitsannotated_combined.counts.csv", header = TRUE)

Counts[Counts == 0] <- NA
# Counts[Counts<30] <- NA
Counts <- na.omit(Counts)
rownames(Counts)<-Counts[,1]
```

```r
Counts[,1] <- NULL

# Counts<-subset(Counts, subset=(GeneID !="NA")) #if no gene symbol, discount

# Countszero <-subset(Counts, subset=(row !=0))
# Countszero <- apply(Counts, 1, function(row) all(row !="NA"))
# Counts <- Counts[Countszero,]

library(limma)
library(edgeR)

Countnum <- Counts[,1:21]
# Counts <- data.matrix(Counts)

# Countnum <- read.csv(file = "pet.counts.clean.csv")

#DGElist
dge <- DGEList(counts=Countnum)
dge <- calcNormFactors(dge)

#Design
Treat<-factor(rep(c("Control", "Patient"),c(8,13)), levels=c("Control", "Patient"))
design<-model.matrix(~Treat)
rownames(design)<-colnames(Countnum)
design

#Voom transformation
v <- voom(dge,design,plot=FALSE)

#Limma fitting
fit <- lmFit(v,design)
fit <- eBayes(fit)
result<-topTable(fit, coef="TreatPatient", adjust="BH", number=nrow(Countnum)) #"BH" adjust for multipl
result <- merge(result, Counts, by="row.names", all=TRUE)
result <- result[,1:7]

#Count tables from bcbio have ensembl gene IDs. This must be annotated with HGNC symbols

#Download the HGNC symbols and gene IDs using a vector containing the IDs from results
library(biomaRt)
genes <- as.vector(result[,1])
mart <- useMart("ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl", host="www.ensembl.org")
mart_back <- getBM(attributes =c("ensembl_gene_id", "hgnc_symbol"), filters="ensembl_gene_id", values=g

# library(org.Hs.eg.db)
# library(GeneNetworkBuilder)

#Merge the tables using ensembl ID
result <- merge(result, mart_back, by.x = "Row.names", by.y = "ensembl_gene_id")
# result[,1] <- NULL

uniqueresult <- result[!duplicated(result$hgnc_symbol),]
rownames(uniqueresult) <- uniqueresult$hgnc_symbol
```

```
genesort <- uniqueresult[order(uniqueresult$adj.P.Val),]

setwd(dir = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
write.csv(genesort, file=paste(analysis.name, "rankeduniqueresult.csv", sep=""), sep="\t", row.names=TRU

# topgene <- genesort[1:1000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_1000.csv", sep = ""))
# topgene <- genesort[1:2000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_2000.csv", sep = ""))
# topgene <- genesort[1:3000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_3000.csv", sep = ""))
# topgene <- genesort[1:4000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_4000.csv", sep = ""))
# topgene <- genesort[1:5000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_5000.csv", sep = ""))
```

Code for Petrucelli:

```
##RNA-Seq Gene Expression Analysis using Limma##

analysis.name<-"PET" #Label analysis
setwd(dir = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/GeneExpressionAnalysis/RNA-seq/Petrucel
# Counts <- read.table(file = 'GSE67196_Petrucelli2015_ALS_genes.rawcount.txt', header = TRUE)
#
# write.csv(x = Counts, file = "counts_petrucelli.csv")

Counts <- read.csv(file = "Pet.annotated_combined.counts.csv", header = TRUE)

Counts[Counts == 0] <- NA
# Counts[Counts<30] <- NA
Counts <- na.omit(Counts)
rownames(Counts)<-Counts[,1]
Counts[,1] <- NULL

# Counts<-subset(Counts, subset=(GeneID !="NA")) #if no gene symbol, discount

# Countszero <-subset(Counts, subset=(row !=0))
# Countszero <- apply(Counts, 1, function(row) all(row !="NA"))
# Counts <- Counts[Countszero,]

library(limma)
library(edgeR)

Countnum <- Counts[,1:27]
# Counts <- data.matrix(Counts)

# Countnum <- read.csv(file = "pet.counts.clean.csv")

#DGElist
dge <- DGEList(counts=Countnum)
dge <- calcNormFactors(dge)

#Design
```

```
Treat<-factor(rep(c("Patient", "Control"),c(18,9)), levels=c("Patient", "Control"))
design<-model.matrix(~Treat)
rownames(design)<-colnames(Countnum)
design

#Voom transformation
v <- voom(dge,design,plot=FALSE)

#Limma fitting
fit <- lmFit(v,design)
fit <- eBayes(fit)
result<-topTable(fit, coef="TreatControl", adjust="BH", number=nrow(Countnum)) #"BH" adjust for multipl
result <- merge(result, Counts, by="row.names", all=TRUE)
result <- result[,1:7]

#Count tables from bcbio have ensembl gene IDs. This must be annotated with HGNC symbols

#Download the HGNC symbols and gene IDs using a vector containing the IDs from results
library(biomaRt)
genes <- as.vector(result[,1])
mart <- useMart("ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl", host="www.ensembl.org")
mart_back <- getBM(attributes =c("ensembl_gene_id", "hgnc_symbol"), filters="ensembl_gene_id", values=g

# library(org.Hs.eg.db)
# library(GeneNetworkBuilder)

#Merge the tables using ensembl ID
result <- merge(result, mart_back, by.x = "Row.names", by.y = "ensembl_gene_id")
# result[,1] <- NULL

uniqueresult <- result[!duplicated(result$hgnc_symbol),]
rownames(uniqueresult) <- uniqueresult$hgnc_symbol
genesort <- uniqueresult[order(uniqueresult$adj.P.Val),]

setwd(dir = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
write.csv(genesort, file=paste(analysis.name, "rankeduniqueresult.csv", sep=""), sep="\t", row.names=TR

# topgene <- genesort[1:1000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_1000.csv", sep = ""))
# topgene <- genesort[1:2000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_2000.csv", sep = ""))
# topgene <- genesort[1:3000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_3000.csv", sep = ""))
# topgene <- genesort[1:4000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_4000.csv", sep = ""))
# topgene <- genesort[1:5000,]
# write.csv(x = topgene, file = paste(analysis.name,"_ap_5000.csv", sep = ""))
```

For the microarray data, I split the ensembl column so as to only take the ensembl ID. I then ran this code for the HGNC consensus and then the ensembl consensus:

```
#QQplot for p Values#
```

```
setwd(dir = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
C9DEgene <- read.csv("C9rankeduniqueresult.csv")
CHDEgene <- read.csv("CHrankeduniqueresult.csv")
sALSDEgene <- read.csv("sALSrankeduniqueresult.csv")
FTLDDEgene <- read.csv("FTLDrankeduniqueresult.csv")
VCPDEgene <- read.csv("VCPrankeduniqueresult.csv")

setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
PETDEgene <- read.csv("PETrankeduniqueresult.csv")
RAVDEgene <- read.csv("RAVrankeduniqueresult.csv")

setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/HG-U133_Plus_2.na35.annot.csv/")
Annotation <- read.csv("HG-U133_Plus_2.na35.annot.csv")

#Create list for C9orf72#
genelist <- C9DEgene$Ensembl
thresh <- c(500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 80
resultC9 <- list(a = 1:500, b = 1:1000, c = 1:1500, d = 1:2000, e = 1:2500, f = 1:3000, g = 1:3500,
                 h = 1:4000, i = 1:4500, j = 1:5000, k = 1:5500, l = 1:6000, m = 1:6500, n = 1:7000,
                 o = 1:7500, p = 1:8000)

for (i in 1:length(thresh)) {
  resultC9[[i]] <- genelist[1:thresh[[i]]]
}

#Create list for CHMP2B#
genelist <- CHDEgene$Ensembl
thresh <- c(500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 80
resultCH <- list(a = 1:500, b = 1:1000, c = 1:1500, d = 1:2000, e = 1:2500, f = 1:3000, g = 1:3500,
                  h = 1:4000, i = 1:4500, j = 1:5000, k = 1:5500, l = 1:6000, m = 1:6500, n = 1:7000,
                  o = 1:7500, p = 1:8000)

for (i in 1:length(thresh)) {
  resultCH[[i]] <- genelist[1:thresh[[i]]]
}

#Create list for sALS#
genelist <- sALSDEgene$Ensembl
thresh <- c(500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 80
resultsALS <- list(a = 1:500, b = 1:1000, c = 1:1500, d = 1:2000, e = 1:2500, f = 1:3000, g = 1:3500,
                  h = 1:4000, i = 1:4500, j = 1:5000, k = 1:5500, l = 1:6000, m = 1:6500, n = 1:7000,
                  o = 1:7500, p = 1:8000)

for (i in 1:length(thresh)) {
  resultsALS[[i]] <- genelist[1:thresh[[i]]]
}

#Create list for FTLD#
genelist <- FTLDDEgene$Ensembl
thresh <- c(500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 80
resultFTLD <- list(a = 1:500, b = 1:1000, c = 1:1500, d = 1:2000, e = 1:2500, f = 1:3000, g = 1:3500,
                  h = 1:4000, i = 1:4500, j = 1:5000, k = 1:5500, l = 1:6000, m = 1:6500, n = 1:7000,
                  o = 1:7500, p = 1:8000)
```

```
for (i in 1:length(thresh)) {
  resultFTLD[[i]] <- genelist[1:thresh[[i]]]
}

#Create list for VCP#
genelist <- VCPDEgene$Ensembl
thresh <- c(500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 80
resultVCP <- list(a = 1:500, b = 1:1000, c = 1:1500, d = 1:2000, e = 1:2500, f = 1:3000, g = 1:3500,
                  h = 1:4000, i = 1:4500, j = 1:5000, k = 1:5500, l = 1:6000, m = 1:6500, n = 1:7000,
                  o = 1:7500, p = 1:8000)

for (i in 1:length(thresh)) {
  resultVCP[[i]] <- genelist[1:thresh[[i]]]
}

#Create list for Petrucelli#
genelist <- PETDEgene$Row.names
thresh <- c(500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 80
resultPET <- list(a = 1:500, b = 1:1000, c = 1:1500, d = 1:2000, e = 1:2500, f = 1:3000, g = 1:3500,
                  h = 1:4000, i = 1:4500, j = 1:5000, k = 1:5500, l = 1:6000, m = 1:6500, n = 1:7000,
                  o = 1:7500, p = 1:8000)

for (i in 1:length(thresh)) {
  resultPET[[i]] <- genelist[1:thresh[[i]]]
}

#Create list for Ravits#
genelist <- RAVDEgene$Row.names.1
thresh <- c(500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 80
resultRAV <- list(a = 1:500, b = 1:1000, c = 1:1500, d = 1:2000, e = 1:2500, f = 1:3000, g = 1:3500,
                  h = 1:4000, i = 1:4500, j = 1:5000, k = 1:5500, l = 1:6000, m = 1:6500, n = 1:7000,
                  o = 1:7500, p = 1:8000)

for (i in 1:length(thresh)) {
  resultRAV[[i]] <- genelist[1:thresh[[i]]]
}

##Intersect lists of same length##
result <- list(list(a = 1:5000, b = 1:5000, c = 1:5000, d = 1:5000, e = 1:5000, f = 1:5000, g = 1:5000,
                    h = 1:5000, i = 1:5000, j = 1:5000, k = 1:5000, l = 1:5000, m = 1:5000, n = 1:5000,
                    o = 1:5000, p = 1:5000))

for (i in 1:16) {
  result[[i]] <- Reduce(intersect, list(resultC9[[i]],resultCH[[i]],resultsALS[[i]],resultFTLD[[i]],
                                        resultVCP[[i]],resultPET[[i]],resultRAV[[i]]))
}

##Make list values equal, convert to data frame and save##
for (i in 1:16) {
  length(result[[i]]) <- 665
}

setwd(dir = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
```

```
resultdf <- as.data.frame(result)
write.csv(result, file = "Ensconsensus.csv")


# result2000 <- as.data.frame(result[[4]])
# result2500 <- as.data.frame(result[[5]])
result3000 <- as.data.frame(result[[6]])
result3500 <- as.data.frame(result[[7]])
result4000 <- as.data.frame(result[[8]])
result4500 <- as.data.frame(result[[9]])
result5000 <- as.data.frame(result[[10]])
result5500 <- as.data.frame(result[[11]])
result6000 <- as.data.frame(result[[12]])
result6500 <- as.data.frame(result[[13]])
result7000 <- as.data.frame(result[[14]])
result7500 <- as.data.frame(result[[15]])
result8000 <- as.data.frame(result[[16]])


#consolidate both hgnc files and ensembl then run this
Z <- read.csv(file = "Allgenes.csv", header = TRUE, na.strings = c("", "NA)"))
Z <- as.list(Z)
Z<- lapply(Z, function(x) x[!is.na(x)])
Z <- lapply(Z, function(x) x[!duplicated(x)])
```

I then ran the fisher's exact test enrichment on the new genes. File is called Allgenes.csv and is in the folder DEG Test2. Again the only signs of enrichment were with the Exac List, ALS gene cards, and the protein protein interaction lists.

I also tried loading some cytoscape modules to analyse the genes. I got Genemania working, but I couldn't get bingo to work or cytokegg which were the two I was most interested in. Cluego is used a lot but I don't care about clustering pathways as much yet, I'd rather cluster the genes into pathways, preferably ones I already have through pathprint.

Tuesday

I found an app for reactome, and inputted all 715 genes from the 8000 consensus. The top 10 results are: Gene Expression Infectious disease Influenza life cycle Influenza infection Influenza viral RNA Disease Translation Peptide chain elongation SRP-dependent cotranslational protein targeting to membrane Nonsense mediated decay enhanced by the exon junction complex

The only two reactome pathways I have in my pathprint list are signaling by insulin receptor and opioid signaling, neither of which reached significance (though these were generated from only microarray data).

After downloading cluepedia, I finally managed to make cluego work. Cluepedia is much better as it allows for using Kegg, Reactome and Wikipathways. I conducted enrichment of my 715 genes, of which 11 kegg pathways were significantly enriched. This are:

Pathways in cancer
Bacterial invasion of epithelial cells
Endocytosis
Huntington's Disease
TGF-beta signaling pathway
mRNA surveillance pathway

Ribosome
Ubiquitin mediated proteolysis
RNA transport
Lysine degredation
FoxO signaling pathway

I then decided to do the same but after expansion from Genemania. I took the 61 genes from 5500 consensus, and expanded to also create 715 genes. (TUG1 was not recognised). I used GO BP weighting, and allowed interaction networks coexpression, genetic interactions and physical interactions. Enrichment was at sig p<.05.

The results are:

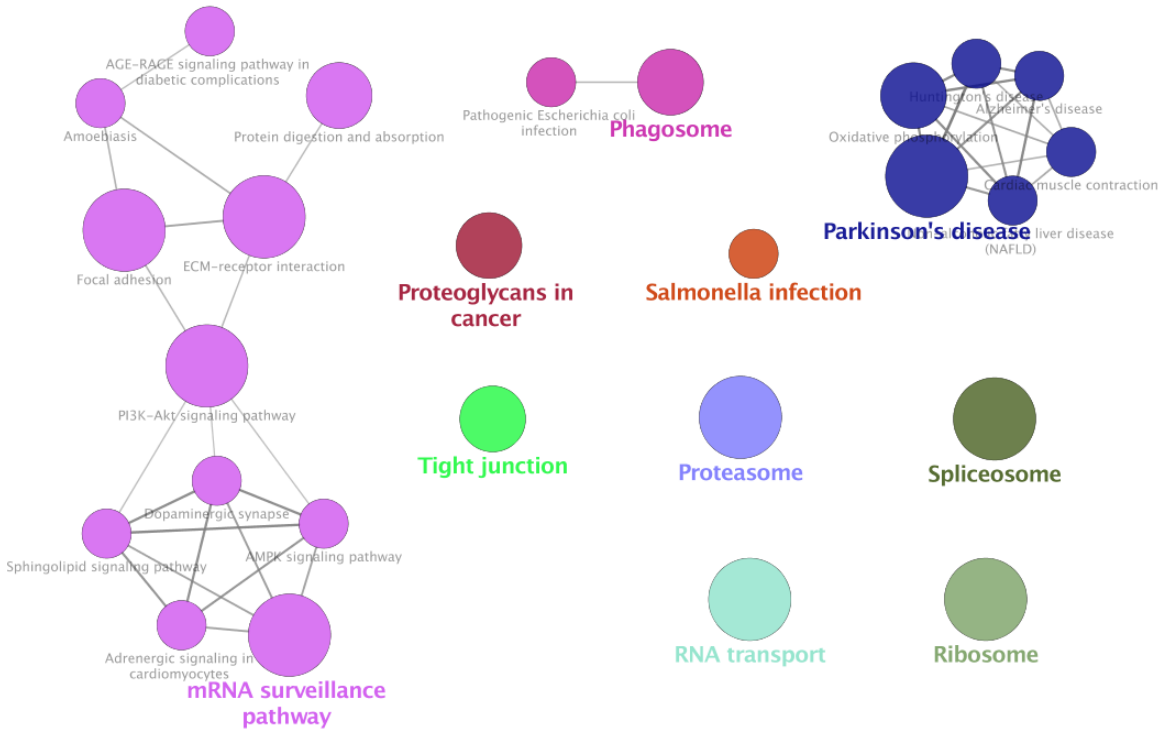

Figure 1:

I am currently trying to divide this into clusters using mcode, but MCODE takes hours, and it keeps crashing. In the mean time, Win suggested conducting WGCNA on each of my datasets and trying to see whether I see common modules coming out.

I'm attempting WGCNA. Annoyingly the example uses phenotype data (which I don't need to use) and it messes up all the code as variables are included which I don't have. I think I have it worked out, and it's as follows:

```
library(WGCNA)
options(stringsAsFactors = FALSE);
### orf72 ###
# Display the current working directory
setwd ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
```

```
Analysis.name <- "C9"

#Read in desired genes
Results <- read.csv ("C9rankeduniqueresult.csv", header=TRUE) #Taking only the genes we deemed acceptab
Results <- Results[1:8000,]
#gene expression analysis to find criteria

ID <- as.data.frame(Results$Probe.Set.ID)
colnames(ID)[1] <- "ProbeID"


#Read in raw expression values
setwd ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/C9orf72_LCM/")
RawExp <- read.csv("HerGlaucrankeduniqueresult.csv")

Exp <- Results
#merge(ID, RawExp, by.x="ProbeID", by.y="Probe.Set.ID") #merge raw expression data with accepted genes
rownames(Exp) <- Exp[,1] #make probeset IDs row names
#colnames(Exp) <- colnames(RawExp) #make file names column names
Exp[,1] <- NULL  #remove ID column
Exp <- Exp[,51:58] #Take only patients

# Pat <- Exp[,1:8]
# Con <- Exp[,9:11]
#
# Pat <- t(Exp)
# Con <- t(Exp)
Exp <- t(Exp)

###PATIENT ANALYSIS###

###Choosing soft threshold
# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to=20, by=2))
# Call the network topology analysis function
sft = pickSoftThreshold(Exp, powerVector = powers, verbose = 5)
# Plot the results:
sizeGrWindow(9, 5)
par(mfrow = c(1,2));
cex1 = 0.9;
# Scale-free topology fit index as a function of the soft-thresholding power
plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
     xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type="n",
     main = paste("Scale independence"));
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
     labels=powers,cex=cex1,col="red");
# this line corresponds to using an R^2 cut-off of h
abline(h=0.50,col="red")
# Mean connectivity as a function of the soft-thresholding power
plot(sft$fitIndices[,1], sft$fitIndices[,5],
     xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n",
     main = paste("Mean connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=cex1,col="red")
```

```
##SOFT THRESHOLD VALUE SELECTED##

# Exp <- data.matrix(Exp) #csv files contain character matrices, the following code requires numeric

##One-step network construction and module detection
setwd ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/DEG Test2/")
net = blockwiseModules(Exp, power = 5,
                       TOMType = "unsigned", minModuleSize = 30,
                       reassignThreshold = 0, mergeCutHeight = 0.25,
                       numericLabels = TRUE, pamRespectsDendro = FALSE,
                       saveTOMs = TRUE,
                       saveTOMFileBase = "TOM",
                       verbose = 3)
table(net$colors)

# open a graphics window
sizeGrWindow(12, 9)

# Convert labels to colors for plotting
mergedColors = labels2colors(net$colors)
# Plot the dendrogram and the module colors underneath
plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
                    "Module colors",
                    dendroLabels = FALSE, hang = 0.03,
                    addGuide = TRUE, guideHang = 0.05)
moduleLabels = net$colors
moduleColors = labels2colors(net$colors)
MEs = net$MEs
geneTree = net$dendrograms[[1]]
save(MEs, moduleLabels, moduleColors, geneTree,
     file = "-networkConstruction-auto.RData")

geneInfo0 = data.frame(ProbeID = Results$Probe.Set.ID,
                       geneSymbol = Results$Gene.Symbol,
                       moduleColor = moduleColors)
rownames(geneInfo0) <- NULL

geneOrder = order(geneInfo0$moduleColor)
geneInfo = geneInfo0[geneOrder, ]

write.csv(geneInfo, file = "C9geneInfo.csv")
```

Thursday

I have completed the above code for all data sets (though sALS and FTLD came out a little weird) Threshold for C9 - 5 CHMP2B - 8 sALS - 10 FTLD - 6 VCP - 7