LabBook_11_03_16

Claire Green

Monday

The first thing I did was download the Read 2 files using Iceberg. While this was downloading, I took a look at Jiantao's code for combining GRAIL and CoXPRESdb to idenitfy the inter-relatedness of genes within a group.

He sent me the following files (all located in the TDP-43_Code Git repository)

GRAIL.R –> an R script containing the functions created for the analysis Grail_CO_TM.R —> The R script which does the actual analysis.

I also ran the read 2 downloads for the petrucelli data. Once this was done, I needed to run bcbio. The first thing to do was to create a .csv file with all the experimental information. For the Petrucelli data, this was constructed as follows:

samplename	description	phenotype	batch
SRR1927020	sample_C9.1	sample_C9	1
SRR1927022	sample_C9.2	sample_C9	1
SRR1927024	sample_C9.3	sample_C9	1
SRR1927026	sample_C9.4	sample_C9	1
SRR1927028	sample_C9.5	sample_C9	1
SRR1927030	sample_C9.6	sample_C9	1
SRR1927032	sample_C9.7	sample_C9	1
SRR1927034	sample_C9.8	sample_C9	1
SRR1927036	sample_sALS.1	sample_sALS	1
SRR1927038	sample_sALS.2	sample_sALS	1
SRR1927040	sample_sALS.3	sample_sALS	1
SRR1927042	sample_sALS.4	sample_sALS	1
SRR1927044	sample_sALS.5	sample_sALS	1
SRR1927046	sample_sALS.6	sample_sALS	1
SRR1927048	sample_sALS.7	sample_sALS	1
SRR1927050	sample_sALS.8	sample_sALS	1
SRR1927052	sample_sALS.9	sample_sALS	1
SRR1927054	sample_sALS.10	sample_sALS	1
SRR1927056	sample_control.1	sample_control	1
SRR1927058	sample_control.2	sample_control	1
SRR1927060	sample_control.3	sample_control	1
SRR1927062	sample_control.4	sample_control	1
SRR1927064	sample_control.5	sample_control	1
SRR1927066	sample_control.6	sample_control	1
SRR1927068	sample_control.7	sample_control	1
SRR1927070	sample_control.8	sample_control	1
SRR1927071	sample_control.9	sample_control	1

The descriptions have to be unique, but the phenotypes are what group samples. Batches are whether your samples have come from the same individual in different batches (e.g. time series data). This .csv file is how bcbio knows how to separate the data into experimental groups, and aids in understanding the output.

This file I made in excel and transferred over. To link to my folder on the server, I went to the Mac "Go"

menu and selected “Connect to server”. Here I inputted the server address “smb://mdp15cmg@uosfstore.shef.ac.uk/shared/hidelab2/user/mdp15cmg” And connected to the TDP-43 folder I had created there. This is now a finder window and I can just drag and drop files into iceberg storage.

The next thing to do is run the script that Sandeep sent to me in terminal:

```
#!/bin/bash
#$ -pe openmp 16
#memory requests are per-core
#$ -l rmem=16G -l mem=16G
#Prefer the hidelab queue but spill over to over queues if it is full
#$ -P hidelab

module load apps/gcc/5.2/bcbio/0.9.6a
work_dir='/shared/hidelab2/user/mdp15cmg/TDP-43/'

#Seq.Reads file directories
tdp43_r1_files=$work_dir/input/Read1
tdp43_r2_files=$work_dir/input/Read2

#Read in seq reads
tdp43_r1=($(find $tdp43_r1_files -type f -name "*.gz"|sort -n))
tdp43_r2=($(find $tdp43_r2_files -type f -name "*.gz"|sort -n))

#Download the best-practice template file for RNAseq experiment
echo "DOWNLOADING TEMPLATE"
bcbio_nextgen.py -w template illumina-rnaseq tdp43_project

#Edit the template
echo "EDITTING TEMPLATE"
#Switch to using star
sed -i 's/tophat2/star/g' $work_dir/tdp43_project/config/tdp43_project-template.yaml

#Initialise the main analysis
echo "INITIALISING ANALYSIS"
bcbio_nextgen.py -w template $work_dir/tdp43_project/config/tdp43_project-template.yaml $work_dir/tdp43

#Perform the analysis
echo "PERFOMING ANALYSIS"
cd $work_dir/tdp43_project/work
bcbio_nextgen.py -n 16 ../config/tdp43_project.yaml
```

First the memory capacity is specified. I ended up needing around 20G rather than 16 but at the moment Iceberg just sends you an email warning you have gone over the limit you specified (this does not mean the process has ended, it’s just a warning). You can check if it is still processing by entering qtop followed by your job number.

Next, it connects to the bcbio module, and sets the working directory. It reads in the two reads and downloads the generic illumina RNA-seq template for the project. You can edit the template to what you want, here we have changed tophat2 to star. Initialising the analysis requires input of the .csv file I made and applies it to each sample in read1 and read2. Finally the analysis is performed. This can take around 2 days to complete.

In the mean time I have been reading up on building networks. My idea is to, in the same way that I have performed my analyses before, build individual networks for each data set (microarray) and detect modules. After enriching for function, I can see what is similar between the 5 data sets. I will have to ask though

whether controls are included. Another idea is to compare cases and controls, though I think this would have to be done individually as the different data sets have batch control problems (and one is a different platform)

Tuesday

On Tuesday I set about trying to understand Jiantao's GRAIL methodology. The files he sent me were thus:

GRAIL.R –> contains all the functions that are used in the analysis GRAIL_CO_TM –> The main R script which is used to run the analysis Query_Genes_New –> the list of genes which are thought to be disease-associated WM_BL_vs_HD_BL –> I believe this is a list of seed genes which are known to be disease-associated

Here is the main code:

```
setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/JiantaoGRAIL")

source("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/JiantaoGRAIL/GRAIL.R")

load("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/JiantaoGRAIL/text_2006_12_Symbol.RData")
load("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/JiantaoGRAIL/Hsa_v12_08_CoExpression_Symbol.R

# read seed genes
sTable = read.table(file = "WM_BL_vs_HD_BL.txt", row.names = 1, header = T, sep = "\t") #table with sig
SeedUP = as.character(sTable[,"Symbol"])[as.numeric(sTable$logFC) > 0] #upregulated genes
SeedDN = as.character(sTable[,"Symbol"])[as.numeric(sTable$logFC) < 0] #downregulated genes
SeedUPDN = c(SeedUP, SeedDN) #gene list ordered by foldchange
SList  = list(SeedUP = unique(SeedUP), SeedDN = unique(SeedDN), SeedUPDN = unique(SeedUPDN)) #take uniq

# query genes
queryVector = as.character(read.table(file = "Query_Genes_New.txt", row.names = NULL, header = F, sep =

# filter by Whole network

for(i in 1:length(SList)){

    SeedVector = as.character(SList[[i]])
    SeedList   = as.list(SeedVector)
    names(SeedList) = SeedVector

    tmList = GrailWorkflow(SeedList, queryVector, TmDB,   method = "coexpression", Nthreads = 4, Pfcuto
    coList = GrailWorkflow(SeedList, queryVector, CoexDB, method = "coexpression", Nthreads = 4, Pfcuto

    # merge results
    tmTable = tmList$qTable
    coTable = coList$qTable
    rownames(tmTable) = as.character(tmTable[,1])
    rownames(coTable) = as.character(coTable[,1])

    ID = unique(c(SeedVector, queryVector))
    NN = length(ID)
    tmBestGene = coBestGene = rep("", NN)
    tmBestPv   = coBestPv   = rep(NA,  NN)
    names(tmBestGene) = names(tmBestPv) = names(coBestGene) = names(coBestPv) = ID
```

```
    tmBestGene[ rownames(tmTable)] = as.character(tmTable[,"queryVector"])
    tmBestPv[ rownames(tmTable)]   = as.numeric(tmTable[,"pvalueVector"])

    coBestGene[ rownames(coTable)] = as.character(coTable[,"queryVector"])
    coBestPv[ rownames(coTable)]   = as.numeric(coTable[,"pvalueVector"])

    Flag   <- ID %in% queryVector
    mTable <- data.frame(ID, tmBestGene, tmBestPv, coBestGene, coBestPv, Flag)

    write.table(mTable, file = paste0("Results/", names(SList)[i], ".txt"), row.names = F, col.names = T
}
```

The method uses two databases - one from COEXPRSdb and another that I'm not entirely sure what it is, but is referred to as TMdb. I need to contact Jiantao to understand what this database is. Both seem to contain coexpression values. I believe the concept of the method is to discover how tightly co-expressed each gene is within the module. However, it is not clear what the role of the seed list and query list are in this calculation. This I will also need to contact Jiantao about.

Wednesday

On Wednesday I mostly did reading and worked a little on my abstract for my presentation.

Thursday

On Thursday I began looking into using WGCNA for creating modules for each of the 5 microarray datasets. Going through the documentation I managed to make it work for my data:

```
library(WGCNA)
options(stringsAsFactors = FALSE);
### C9orf72 ###
# Display the current working directory
setwd ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/GeneExpressionAnalysis/Microarray/TopGenes_1

Analysis.name <- "C9"

#Read in desired genes
C9Results <- read.csv ("C9rankeduniqueresult.csv", header=TRUE) #Taking only the genes we deemed accept
#gene expression analysis to find criteria

C9ID <- as.data.frame(C9Results$Probe.Set.ID)
colnames(C9ID)[1] <- "ProbeID"


#Read in raw expression values
setwd ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/C9orf72_LCM/")
C9RawExp <- read.csv("eset_NineP_150612_exprs.csv")

C9Exp <-merge(C9ID, C9RawExp, by.x="V1", by.y="X") #merge raw expression data with accepted genes
rownames(C9Exp) <- C9Exp[,1] #make probeset IDs row names
colnames(C9Exp) <- colnames(C9RawExp) #make file names column names
```

```
C9Exp <- cbind(C9Exp[,2:12]) #remove ID column

# C9Pat <- C9Exp[,1:8]
# C9Con <- C9Exp[,9:11]
#
# C9Pat <- t(C9Exp)
# C9Con <- t(C9Exp)
C9Exp <- t(C9Exp)

###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(C9Pat, 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=0.70,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 OF 6 SELECTED##

# C9Exp <- data.matrix(C9Exp) #csv files contain character matrices, the following code requires numeri

##One-step network construction and module detection
setwd ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Data/WGCNA/C9orf72/PatCon/")
net = blockwiseModules(C9Pat, power = 6,
                       TOMType = "unsigned", minModuleSize = 30,
                       reassignThreshold = 0, mergeCutHeight = 0.25,
                       numericLabels = TRUE, pamRespectsDendro = FALSE,
                       saveTOMs = TRUE,
                       saveTOMFileBase = "C9TOM",
                       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 = "C9-networkConstruction-auto.RData")


### RELATING MODULES TO EXTERNAL INFORMATION AND IDENTIFYING IMPORTANT GENES ###
#lnames = load(file = "C9-networkConstruction-auto.RData")

traitdata <- read.csv(file = "C9Pheno.csv")
Samples = rownames(C9Exp)
traitRows = match(Samples, traitdata$SampleName)
datTraits = traitdata[traitRows, -1]
rownames(datTraits) = datTraits[traitRows, 1]
Traitpheno <- datTraits[,2]

# Define numbers of genes and samples
nGenes = ncol(C9Exp)
nSamples = nrow(C9Exp)
# Recalculate MEs with color labels
MEs0 = moduleEigengenes(C9Exp, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, Traitpheno, use = "p")
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples)


sizeGrWindow(10,6)
# Will display correlations and their p-values
textMatrix = paste(signif(moduleTraitCor, 2), "(", signif(moduleTraitPvalue, 1), ")", sep = " ");
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(6, 8.5, 3, 3))
# Display the correlation values within a heatmap plot
labeledHeatmap(Matrix = moduleTraitCor,
               xLabels = "Disease Phenotype",
               yLabels = names(MEs),
               ySymbols = names(MEs),
               colorLabels = FALSE,
               colors = greenWhiteRed(50),
               textMatrix = textMatrix,
               setStdMargins = FALSE,
               cex.text = 0.9,
               zlim = c(-1,1),
               main = paste("Module-trait relationships"))
```

```
### Gene relationship to trait and important modules: Gene Significance and Module Membership ##

# Define variable weight containing the weight column of datTrait
pheno = as.data.frame(datTraits$PhenoLabel)
names(pheno) = "Disease Phenotype"
# names (colors) of the modules
modNames = substring(names(MEs), 3)
geneModuleMembership = as.data.frame(cor(C9Exp, MEs, use = "p"))
MMPvalue = as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership), nSamples))
names(geneModuleMembership) = paste("MM", modNames, sep="")
names(MMPvalue) = paste("p.MM", modNames, sep="")
geneTraitSignificance = as.data.frame(cor(C9Exp, pheno, use = "p"))
GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance), nSamples))
names(geneTraitSignificance) = paste("GS.", names(pheno), sep="")
names(GSPvalue) = paste("p.GS.", names(pheno), sep="")

module = "brown"
column = match(module, modNames)
moduleGenes = moduleColors==module
sizeGrWindow(7, 7)
par(mfrow = c(1,1))
verboseScatterplot(abs(geneModuleMembership[moduleGenes, column]),
                   abs(geneTraitSignificance[moduleGenes, 1]),
                   xlab = paste("Module Membership in", module, "module"),
                   ylab = "Gene significance for disease phenotype",
                   main = paste("Module membership vs. gene significance\n"),
                   cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col = module)


#find out what the IDs are of the genes that are contained within a module.

colnames(C9Exp)[moduleColors=='green']


geneInfo0 = data.frame(ProbeID = C9Results$Probe.Set.ID,
                       geneSymbol = C9Results$Gene.Symbol,
                       moduleColor = moduleColors,
                       geneTraitSignificance,
                       GSPvalue)
rownames(geneInfo0) <- NULL

modOrder = order(-abs(cor(MEs, pheno, use = "p")));
# Add module membership information in the chosen order
for (mod in 1:ncol(geneModuleMembership))
{
  oldNames = names(geneInfo0)
  geneInfo0 = data.frame(geneInfo0, geneModuleMembership[, modOrder[mod]],
                         MMPvalue[, modOrder[mod]]);
  names(geneInfo0) = c(oldNames, paste("MM.", modNames[modOrder[mod]], sep=""),
                       paste("p.MM.", modNames[modOrder[mod]], sep=""))
}
# Order the genes in the geneInfo variable first by module color, then by geneTraitSignificance
geneOrder = order(geneInfo0$moduleColor, -abs(geneInfo0$GS.Disease.Phenotype))
```

```
geneInfo = geneInfo0[geneOrder, ]

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


### Interfacing network analysis with other data such as functional annotation and gene ontology ###

## Output gene lists for use with online software and services

C9Entrez <- cbind(C9Results$Probe.Set.ID, C9Results$Entrez.Gene)
colnames(C9Entrez)[1] <- "ProbeID"
colnames(C9Entrez)[2] <- "EntrezID"

geneInfo <- merge(geneInfo, C9Entrez, by.x="ProbeID", by.y="ProbeID")

AllIDs <- geneInfo$EntrezID
# # $ Choose interesting modules
# intModules = c("royalblue", "brown", "green", "purple", "greenyellow")
# for (module in intModules)
# {
#   # Select module probes
#   modGenes = (moduleColors==module)
#   # Get their entrez ID codes
#   modLLIDs = IDs[modGenes];
#   # Write them into a file
#   fileName = paste("IDs-", module, ".txt", sep="");
#   write.table(as.data.frame(modLLIDs), file = fileName,
#               row.names = FALSE, col.names = FALSE)
# }
# # As background in the enrichment analysis, we will use all probes in the analysis.
# fileName = paste("IDs-all.txt", sep="");
# write.table(as.data.frame(IDs), file = fileName,
#             row.names = FALSE, col.names = FALSE)

##Enrichment analysis directly within R
#source("https://bioconductor.org/biocLite.R")
#biocLite("org.Hs.eg.db")

intModules <- c("green", "purple", "brown", "greenyellow", "royalblue")

GOenr = GOenrichmentAnalysis(moduleColors, AllIDs, ontologies = c("BP"), organism = "human", nBestP = 1

tab = GOenr$bestPTerms[[4]]$enrichment

write.table(tab, file = "C9GOEnrichmentTableBP.csv", sep = ",", quote = TRUE, row.names = FALSE)


###Exporting to Cytoscape

# Calculate topological overlap anew: this could be done more efficiently by saving the TOM
# calculated during module detection, but let us do it again here.
intModules <- "brown"
```

```
TOM = TOMsimilarityFromExpr(C9Exp, power = 6);
probes = colnames(C9Exp)
inModule = is.finite(match(moduleColors, intModules));
modProbes = probes[inModule];
modGenes = geneInfo$geneSymbol [match(modProbes, geneInfo$ProbeID)];
modTOM = TOM[inModule, inModule];
dimnames(modTOM) = list(modProbes, modProbes)
# Export the network into edge and node list files Cytoscape can read
cyt = exportNetworkToCytoscape(modTOM,
                               edgeFile = paste("CytoscapeInput-edges-", paste(intModules, collapse="-")
                               nodeFile = paste("CytoscapeInput-nodes-", paste(intModules, collapse="-")
                               weighted = TRUE,
                               threshold = 0.02,
                               nodeNames = modProbes,
                               altNodeNames = modGenes,
                               nodeAttr = moduleColors[inModule]);



# dissTOM = 1-TOMsimilarityFromExpr(C9Exp, power = 6);
# # Transform dissTOM with a power to make moderately strong connections more visible in the heatmap
# plotTOM = dissTOM^7;
# # Set diagonal to NA for a nicer plot
# diag(plotTOM) = NA;
#
# # Call the plot function
# sizeGrWindow(9,9)
# TOMplot(plotTOM, moduleColors, main = "Network heatmap plot, all genes")
#
# # Recalculate module eigengenes
# MEs = moduleEigengenes(datExpr, moduleColors)$eigengenes
# # Isolate weight from the clinical traits
# weight = as.data.frame(datTraits$weight_g);
# names(weight) = "weight"
# # Add the weight to existing module eigengenes
# MET = orderMEs(cbind(MEs, weight))
# # Plot the relationships among the eigengenes and the trait
# sizeGrWindow(5,7.5);
# par(cex = 0.9)
# plotEigengeneNetworks(MET, "", marDendro = c(0,4,1,2), marHeatmap = c(3,4,1,2), cex.lab = 0.8, xLabel
#                       = 90)
```

My method was thus: 1) Take all genes that had passed through filtering that was done for DEG analysis 2) Conduct network construction and module detection 3) Relate modules to phenotypic metadata and identify important genes 4) Correlate genes' degree of module membership with association with trait 5) Conduct GO Enrichment analysis 6) Export to cytoscape

Friday

Though I got the WGCNA analysis to work, I have some reservations:

1) Should I be combining datasets? Creating a combined network is probably of no use. I can create

individual networks for disease vs control but that poses two problems:

a) How do you compare WGCNA networks?
b) What is the minimum number of samples needed to make a network you are confident in? E.g. C9orf72 data set only has 3 controls. b.2) if I have to accumulate cases and controls, how do I do that when I have a batch effect?

I need to think about what question I am trying to answer, how theoretically to answer that question, and then realistically what methods I can use to answer that question.