Lab Book 20_11_15

Claire Green

Monday, Tuesday, Wednesday

In the first part of the week I started setting up the GSEA using the RGSEA script. This turned out to be more complicated than I first thought because 1) I have never worked off an imported script file before and 2) the script John gave me turned out to need a bit of modifying.

The first problem I had was getting the import of the ensembl database information appropriate for my data. I was using Biomart to do this, and as it turns out, Biomart are having some issues and the info is now being managed by ensembl.org. So the first part is:

```
setwd ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/") #set working directory
CHMP2B_exp <- read.csv("eset_CHMP2B_250615_exprs.csv") #import expression file
row.names(CHMP2B_exp) <- CHMP2B_exp[,1] #make Probe IDs row names if not already done
CHMP2B_exp[,1] <- NULL #remove column containing probe IDs

library (WGCNA)
options(stringsAsFactors = FALSE)

DATA <- CHMP2B_exp
DATA <- t(DATA) #transpose data (this is required for Biomart)


####GSEA

### connect to the biomart database that is correct for your data ###
library (biomaRt)
mart <- useMart("ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl", host="www.ensembl.org")
x <- colnames(DATA) #create vector containing probe IDs


### Retrive the specified attributes from the above database ###

# mart_attribute <- listAttributes(mart)
# mart_filter <- listFilters(mart)

# mart_back <- getBM(attributes =c("hgnc_symbol", "entrezgene", "ensembl_transcript_id"),
#                    filters="ensembl_transcript_id", values=x,  mart=mart)

### Create array of attributes required for your data set ###
mart_back <- getBM(attributes=c("hgnc_symbol", "entrezgene", "ensembl_transcript_id","affy_hg_u133_plus
                   filters = "affy_hg_u133_plus_2", values = x, mart = mart)

e2 <- mart_back[,1] #take hgnc label column
e3 <- which(!(e2 == "")) #remove any rows with blank cells
mart_back1 <- mart_back[c(e3),] #apply to data frame
```

Now you have a list of hgnc labels for your genes. The next stage is to replace the probe IDs in your data set with the gene labels

```
### Assign hgnc symbol as column names of expression data ###
#This looks at the ID probes from the expression file and corresponds them to the list
#ensembl_transcript_ID. The expression file column names are then changed to the corresponding
#hgnc symbol

#This will take a decent amount of time, so don't worry if it runs for 30 minutes+

DATA1 <- DATA

for (i in 1:length(mart_back1[,4]))
{
c1 <- which (x %in% mart_back1[i,4])
c1 <- c1[1]
#colnames (DATA1)[c(c1)] <- mart_back1[i,2]
colnames (DATA1)[c(c1)] <- mart_back1[i,1]
}
#t <- array (dim =c(length(DATA[,1]), length (DATA[1,]), d))
DATA1b <- t(DATA1) #transpose back
write.csv(x = DATA1b, file = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/CHMP2B_ex
```

Thursday

So far I had been managing by creating my gct files externally in excel and then using them in the GSEA code. However, a problem came about when I realised that when you assign gene names to probes you get multiple probes per gene. This meant that I had no good way of either normalising across the probes or picking a suitable representative probe. I decided then to try and understand a section of code given to me by John that creates the gct file in the correct format.

```
t2 <- DATA1 #t2 needs samples as row names and gene IDs as column names

# t1 <- grep("ENST", colnames(t)) #This is for when you are using data with ensembl transcript IDs
# t2 <- t [,-c(t1)]

t1a <- which (colnames(t2) %in% "") #take column names in t2 that contain blanks
if (length (t1a)>0) {t2 <- t2 [,-c(t1a)]} #if there are any blanks, remove them (this will be 0 if you

t2.colnames <- toupper(colnames(t2)) #convert all column names of t2 to uppercase

#create an empty matrix with same rows as t2, but with only the same number of columns as unique names
t2.agg <- matrix (nrow=length(t2[,1]), ncol=length(unique(t2.colnames)))
row.names (t2.agg) <- row.names (t2) #make the row names the same as t2
colnames (t2.agg) <- sort(unique(t2.colnames)) #make the colum names = to unique t2 column names and so

#I think this aggregates the t2 columns by name, takes the mean, and then applies it back to the data f
for (i in 1:length (t2[,1]))
{
agg <- aggregate(t2[i,],by=list(t2.colnames),mean)
t2.agg[i,] <- agg[,2]
}
```

You then have to bind this information into a .gct file containing your expression data and the corresponding hgnc gene labels

```
j="All."
t.agg1 <- t(t2.agg)
t.agg2 <- cbind (row.names(t.agg1), rep(NA,length(t.agg1[,1])), t.agg1)
colnames(t.agg2)[1:2] <- c("NAME", "DESCRIPTION")
#t.agg2 <- data.matrix(t.agg2)
#mode(t.agg2) <- "numeric"
write("#1.2", file = paste("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/", j, "CHMP
write(paste(dim(t.agg2)[1], dim(t.agg1)[2], sep = "\t"),file = paste("/Users/clairegreen/Documents/PhD/
write.table (t.agg2, file= paste("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/", j,
```

Next you have to make your cls file - this contains meta-data on the phenotype groups you have in your data set

```
#Create cls file
write(paste("10", "2", "1", sep = " "), file = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets
write(paste("#", "PAT", "CON", sep = " "),file = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data S
write(paste("0","0","0","1","1","1","1","1","1", sep = " "),file = "/Users/clairegreen/Documents/PhD/TD
```

Finally, you load the GSEA script file, and run the analysis

```
GSEA.program.location <- ("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/GSEA-P-R/GSEA.1.0.R
source(GSEA.program.location, verbose=T, max.deparse.length=9999)


# dir.create(paste("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/C9orf72_LCM/GSEA_output"))

GSEA(                                                              # Input/Output Files :--------
 input.ds =  paste("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/All.CHMP2B.gct"), #
 input.cls = "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/CHMP2B_pheno.cls", # Inpu
 gs.db =     "/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/GSEA-P-R/GeneSetDatabases/c5.bp.v
 output.directory      = paste("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/GSEA2/"
#  Program parameters :-----------------------------------------------------------------------------
 doc.string            = "CHMP2B",   # Documentation string used as a prefix to name result files (defa
 non.interactive.run   = F,               # Run in interactive (i.e. R GUI) or batch (R command line) m
 reshuffling.type      = "sample.labels", # Type of permutation reshuffling: "sample.labels" or "gene.l
 nperm                 = 1000,           # Number of random permutations (default: 1000)
 weighted.score.type   =  1,              # Enrichment correlation-based weighting: 0=no weight (KS), 1
 nom.p.val.threshold   = -1,              # Significance threshold for nominal p-vals for gene sets (de
 fwer.p.val.threshold  = -1,              # Significance threshold for FWER p-vals for gene sets (defau
 fdr.q.val.threshold   = 0.25,            # Significance threshold for FDR q-vals for gene sets (defaul
 topgs                 = 20,              # Besides those passing test, number of top scoring gene sets
 adjust.FDR.q.val      = F,               # Adjust the FDR q-vals (default: F)
 gs.size.threshold.min = 15,              # Minimum size (in genes) for database gene sets to be consid
 gs.size.threshold.max = 500,             # Maximum size (in genes) for database gene sets to be consid
 reverse.sign          = F,               # Reverse direction of gene list (pos. enrichment becomes neg
 preproc.type          = 0,               # Preproc.normalization: 0=none, 1=col(z-score)., 2=col(rank)
 random.seed           = 3338,            # Random number generator seed. (default: 123456)
 perm.type             = 0,               # For experts only. Permutation type: 0 = unbalanced, 1 = bal
 fraction              = 1.0,             # For experts only. Subsampling fraction. Set to 1.0 (no resa
 replace               = F,               # For experts only, Resampling mode (replacement or not repla
 save.intermediate.results = F,           # For experts only, save intermediate results (e.g. matrix of
 OLD.GSEA              = F,               # Use original (old) version of GSEA (default: F)
 use.fast.enrichment.routine = T          # Use faster routine to compute enrichment for random permuta
```

```
)
#-----------------------------------------------------------------------------------------------

results0 <- matrix(unlist(read.table("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/GS
                   ncol=max(count.fields("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP
results1 <- matrix(unlist(read.table("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP2B/GS
                   ncol=max(count.fields("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43 Data Sets/CHMP

print (results0[1:10,])
print (results1[1:10,])
```