Stepmber W1

Awais Choudhry

05/09/2017

## Conducting Gene Ontology

Just doing a simple gene ontology of enhancer-Promoter interactions in R using Goanan.

I opted to use the EnrichR gui as it was used previously with Tessa's PolyA RNA-seq.

but none the less its still good to see what genes are enriched.

library(limma)  
library(biomaRt)  
library(readr)  
library(knitr)  
  
EnhancerPromoterInteractions <- read\_delim("~/Thesis/EnhancerPromoterInteractions",   
 " ", escape\_double = FALSE, trim\_ws = TRUE)  
  
# DataFrameRNASeqIds<-do.call(rbind.data.frame,strsplit(EnhancerPromoterInteractions$`Gene Symbol UCSC Transcript Id Chromosome Start End Enhancer Q\_value`,split = " "))  
write.table(EnhancerPromoterInteractions, quote = FALSE, append= FALSE, row.names= FALSE, sep= "\t",  
 file = "~/Thesis/EnhancerPromoterInteractionsHuman")  
  
human\_mart <- biomaRt::useMart("ENSEMBL\_MART\_ENSEMBL", dataset="hsapiens\_gene\_ensembl")  
ucscToEntrez <- getBM(attributes = c("ucsc", "entrezgene", "external\_gene\_name"),  
 filters = "ucsc",  
 values = list("ucsc" = EnhancerPromoterInteractions$Symbol), mart = human\_mart)  
uniservser <-getBM(attributes = c("ucsc", "entrezgene", "external\_gene\_name"), mart = human\_mart)  
  
  
#######################################33  
###Here we are just lifting over the PoylA genes to Human orthologues but this did not work because biomart only retunred 73 of the 825  
########################################  
  
# # Basic function to convert human to mouse gene names  
# convertHumanGeneList <- function(x){  
#   
# require("biomaRt")  
# human = useMart("ensembl", dataset = "hsapiens\_gene\_ensembl")  
# mouse = useMart("ensembl", dataset = "mmusculus\_gene\_ensembl")  
#   
# genesV2 = getLDS(attributes = c("hgnc\_symbol", "ucsc"), filters = "hgnc\_symbol",  
# values = Supplementary\_tables\_1\_3$`Gene ID`,mart = human, attributesL = c("hsapiens\_homolog\_ensembl\_gene", "ensembl\_gene\_id"), martL = mouse, uniqueRows=T)  
#   
#   
# # Print the first 6 genes found to the screen  
# print(head(genesV2))  
# return(genesV2)  
# }  
#   
  
genes <- convertHumanGeneList(EnhancerPromoterInteractions$`Gene Symbol`)  
  
write.table(genesV2, file= "~/DataFiles/RNAseq/Human/LiftOverPolyADereg",  
 row.names=FALSE,  
 quote=FALSE,  
 append=FALSE)  
# genes<-cbind.data.frame("genes"=genes)  
  
  
  
library(readxl)  
library(dplyr)  
Supplementary\_tables\_1\_3 <- read\_excel("~/DataFiles/Tessa differential Expressed/Supplementary\_tables\_1-3.xls",   
 sheet = "Table 1- PA1 deregulated gene",   
 skip = 1)  
  
RNAseqValidated<-left\_join(x =genes, y= Supplementary\_tables\_1\_3,by= c("genes"="Gene ID") )  
  
HumanValidatedGenes<-filter(RNAseqValidated, !is.na(RNAseqValidated$logFC))  
  
  
  
  
  
OntologyTerms <-goana(de= ucscToEntrez$entrezgene, universe = uniservser$entrezgene, FDR=0.05)  
SignificantGoTerms <-subset(OntologyTerms, P.DE<=0.05)  
OrderedDataTable<-SignificantGoTerms[order(SignificantGoTerms$DE, decreasing = TRUE),]  
OrderedDataTable%>%kable(caption = "List of significant Go terms relating to the genes interacting with ARX enriched Enhancers")

### This is some very messsy and confusing code to get the promoter and enhancer list from the GI object

so the issue is that in ecah row there is an enhancer and a list of promoters. So i wanted to have 1 enhanecr-promoter per row. To do this, i had to get tricky and fly with my transformations. but basically i subset out each row, and replicate the enhancer the length of the list of promoters and then bind them back together. Boy this took me ages.

Oh and i add gene names back to it.

# EDIT: this wont run as i have changed the genome object: see thesis code for updated clearer version of getting the enhancer promoter code

# hippocampuschromhmm<-import("~/DataFiles/ChromHMM/human/E071\_15\_coreMarks\_dense.bed.gz")  
# fetalThymushmm<-import("~/DataFiles/ChromHMM/human/E087\_15\_coreMarks\_dense.bed.gz")  
# PancreasIsletsHMM<-import("~/DataFiles/ChromHMM/human/E093\_15\_coreMarks\_dense.bed.gz")  
# PancreaseHMM<-import("~/DataFiles/ChromHMM/human/E098\_15\_coreMarks\_dense.bed.gz")  
# stomacheMuscleHMM<-import("~/DataFiles/ChromHMM/human/E111\_15\_coreMarks\_dense.bed.gz")  
# fetalBrainMaleHMM<-import("~/DataFiles/ChromHMM/human/E081\_15\_coreMarks\_dense.bed.gz")  
# fetalBrainFemaleHMM<-import("~/DataFiles/ChromHMM/human/E082\_15\_coreMarks\_dense.bed.gz")  
# liverHMM<- import("~/DataFiles/ChromHMM/human/E066\_15\_coreMarks\_dense.bed.gz")  
# H9NeuronCellsHMM<-import("~/DataFiles/ChromHMM/human/E010\_15\_coreMarks\_dense.bed.gz")  
# H9NeuronProgenitorCellsHMM<-import("~/DataFiles/ChromHMM/human/E009\_15\_coreMarks\_dense.bed.gz")  
#   
#   
# celltypesList<-c("HippoCampus"=hippocampuschromhmm,  
# "Pancreatic Islets"= PancreasIsletsHMM,  
# "Fetal Thymus"= fetalThymushmm,  
# "Fetal Brain Male"= fetalBrainMaleHMM,  
# "Fetal Brain, Female"= fetalBrainFemaleHMM,  
# "H9 Neuron Cells"=H9NeuronCellsHMM,  
# "H9 Neuronal Progenitor Cells"= H9NeuronProgenitorCellsHMM,  
# "Pancreas" = PancreaseHMM)  
   
  
Spilt1<-subset(Counts, !promotersIds1=="NA")  
  
  
  
# promotersMerged<-unite(Counts, "Promoters", c("promotersIds1", "promoterIds2"), remove=TRUE)  
# enhancersMerged<-unite(promotersMerged, "Enhancers", c("enhancersIds1", "enhancerIds2"), remove=TRUE)  
# gsub("\\(\\)\\.\_", "", enhancersMerged$Promoters)  
#   
# gsub("^.\*?\_","",enhancersMerged$Promoters)  
# dataFrame<-separate(enhancersMerged, col = "Promoters",into = c("Promoters") )  
  
  
Spilt1<-cbind.data.frame("PromoterIds"=Spilt1$promotersIds1,  
 "EnhancerIds"= Spilt1$enhancerIds2,  
 "Counts" = Spilt1$counts,   
 "q\_value" = Spilt1$q\_value)   
  
Split2<-cbind.data.frame("PromoterIds"=Split2$promoterIds2,  
 "EnhancerIds"= Split2$enhancersIds1,  
 "Counts" = Split2$counts ,   
 "q\_value" = Split2$q\_value)   
  
PromoterIds<-rbind(Spilt1,Split2)  
  
# human\_mart <- biomaRt::useMart("ENSEMBL\_MART\_ENSEMBL", dataset="hsapiens\_gene\_ensembl")  
#   
# ucscToEntrez <- getBM(attributes = c("ucsc", "entrezgene", "external\_gene\_name"),  
# filters = "ucsc",  
# values = list("ucsc" =list ),   
# mart = human\_mart)  
  
# ucscToEntrez$ucsc<-as.character(ucscToEntrez$ucsc)  
  
PromoterIds$PromoterIds<-lapply(PromoterIds$PromoterIds, list)  
  
  
PromoterIds$PromoterIds<-lapply(PromoterIds$PromoterIds,function(x){do.call(rbind, x)})  
  
  
  
dataFrame2<-NULL  
for(i in 1:dim(PromoterIds)[1]){  
test<-PromoterIds[i,]  
  
dataFrame1<-NULL  
for(t in 1:(test$PromoterIds%>%as.data.frame%>%dim)[2]){  
 dataFrame1<-rbind(dataFrame1,test[2:4])  
}  
  
dataFrame2<-rbind.data.frame(dataFrame2,  
 cbind.data.frame("Promoter Ids" =t(test$PromoterIds%>%as.data.frame()),  
 dataFrame1)  
)  
}  
dataFrame2$`Promoter Ids`<-dataFrame2$`Promoter Ids`%>%as.character()  
#   
# human\_mart <- biomaRt::useMart("ENSEMBL\_MART\_ENSEMBL", dataset="hsapiens\_gene\_ensembl")  
#   
# ucscToEntrez <- getBM(attributes = c("ucsc", "entrezgene", "external\_gene\_name"),  
# filters = "ucsc",  
# values = list("ucsc" =dataFrame2$`Promoter Ids`),   
# mart = human\_mart)  
# uniservser <-getBM(attributes = c("ucsc", "entrezgene", "external\_gene\_name"), mart = human\_mart)  
# dataFrame2$`Promoter Ids`<-dataFrame2$`Promoter Ids`%>%as.character  
  
  
library(readr)  
library(dplyr)  
UCSCConvter <- read\_delim("~/DataFiles/Gene Tracks/Human/hg19WithNames.bed",   
 "\t", escape\_double = FALSE, trim\_ws = TRUE)  
  
  
GeneSymbolTHesisTable<-left\_join(dataFrame2, UCSCConvter, by= c("Promoter Ids" = "hg19.kgXref.kgID" ))  
  
EnhancerPromoterInteractionScoreCounts<-cbind.data.frame("Gene Symbol"=GeneSymbolTHesisTable$hg19.kgXref.geneSymbol,  
 "UCSC Transcript Id"= GeneSymbolTHesisTable$`Promoter Ids`,  
 "Chromosome"= GeneSymbolTHesisTable$hg19.knownGene.chrom,  
 "Start" = GeneSymbolTHesisTable$hg19.knownGene.txStart,  
 "End" =GeneSymbolTHesisTable$hg19.knownGene.txEnd,  
 "Enhancer"=GeneSymbolTHesisTable$EnhancerIds,   
 "Q\_value"=GeneSymbolTHesisTable$q\_value)  
  
  
  
  
  
ActiveGenes<-lapply(ActiveInteractionsBasedOnCellType, InteractionPromoterEnhancerInteraction)

### Meeting with steve regrading the ChIP-chip data

So i intially used the Ringo packaage and a gui to analyse the ChIP-chip data of Quille et al (2011) for ARX however, Steve, being much more experienced than i with ChIP-chip suggested to use the limma package as its more evident about what it is doing at each step.

Upon further investigation of the data, we found it is heaviyl skewed to the red so thats alarming. Hence lowess can tolerate upto 30% skewing we have to use this method, also its consistent with what Quille used.

So we are going to treat it like a 2 color probe array and identify differentially expressed probes that increase expression.

Below is the code we generated during the meeting to investigate the data, but to various changes to my file formats and what not it will not run but i used it to generate the limma pipeline.

# lapply(1:3, function(i){MASS::rlm(x~y, data = data.frame(x = log2(RG$G[,i]), y = log2(RG$R[,i])))})  
   
  
# lapply(1:3, function(i){MASS::rlm(x~y, data = data.frame(x = log2(RG$G[,i]), y = log2(RG$R[,i])))})  
# [[1]]  
# Call:  
# rlm(formula = x ~ y, data = data.frame(x = log2(RG$G[, i]), y = log2(RG$R[,   
# i])))  
# Converged in 4 iterations  
#   
# Coefficients:  
# (Intercept) y   
# -0.1870973 0.9196330   
#   
# Degrees of freedom: 243496 total; 243494 residual  
# Scale estimate: 0.307   
#   
# [[2]]  
# Call:  
# rlm(formula = x ~ y, data = data.frame(x = log2(RG$G[, i]), y = log2(RG$R[,   
# i])))  
# Converged in 4 iterations  
#   
# Coefficients:  
# (Intercept) y   
# -0.1492278 0.9545486   
#   
# Degrees of freedom: 243496 total; 243494 residual  
# Scale estimate: 0.289   
#   
# [[3]]  
# Call:  
# rlm(formula = x ~ y, data = data.frame(x = log2(RG$G[, i]), y = log2(RG$R[,   
# i])))  
# Converged in 4 iterations  
#   
# Coefficients:  
# (Intercept) y   
# -0.1680587 0.9386531   
#   
# Degrees of freedom: 243496 total; 243494 residual  
# Scale estimate: 0.162   
  
lapply(1:3, function(i){MASS::rlm(y~x, data = data.frame(x = log2(RG$G[,i]), y = log2(RG$R[,i])))})  
# [[1]]  
# Call:  
# rlm(formula = y ~ x, data = data.frame(x = log2(RG$G[, i]), y = log2(RG$R[,   
# i])))  
# Converged in 4 iterations  
#   
# Coefficients:  
# (Intercept) x   
# 0.7887626 1.0114963   
#   
# Degrees of freedom: 243496 total; 243494 residual  
# Scale estimate: 0.328   
#   
# [[2]]  
# Call:  
# rlm(formula = y ~ x, data = data.frame(x = log2(RG$G[, i]), y = log2(RG$R[,   
# i])))  
# Converged in 4 iterations  
#   
# Coefficients:  
# (Intercept) x   
# 0.5695363 0.9972165   
#   
# Degrees of freedom: 243496 total; 243494 residual  
# Scale estimate: 0.3   
#   
# [[3]]  
# Call:  
# rlm(formula = y ~ x, data = data.frame(x = log2(RG$G[, i]), y = log2(RG$R[,   
# i])))  
# Converged in 4 iterations  
#   
# Coefficients:  
# (Intercept) x   
# 0.3467194 1.0431518   
  
# Degrees of freedom: 243496 total; 243494 residual  
# Scale estimate: 0.173   
  
 plot(log2(RG$G[,1]), log2(RG$R[,1]), pch = 16, cex = 0.6)  
 abline(a= -0.6659529 , b = 1.1071585 , col = "blue")  
 MA <- normalizeWithinArrays(RG, method = "loess")  
  
#An object of class "MAList"  
# $targets  
#   
# $M  
#   
  
 boxplot(MA$M)  
 plotMA(MA)  
 par(mfrow =c(2, 1))  
 plotMA(MA, 1)  
 par(mfrow =c(2, 2))  
 plotMA(MA, 1)  
 plotMA(MA, 2)  
 plotMA(MA, 3)  
 boxplot(MA)  
# Error in sort.int(x, na.last = na.last, decreasing = decreasing, ...) :   
 # 'x' must be atomic  
 par(mfrow =c(1,1))  
 boxplot(MA$M)  
 boxplot(MA$A)  
 boxplot(MA$A, ylim = c(6, 9))  
 boxplot(MA$A, ylim = c(6.5, 9.5))  
 MAq <- normalizeBetweenArrays(MA, method = "Aquantile")  
 boxplot(MAq$A, ylim = c(6.5, 9.5))  
 par(mfrow =c(2, 2))  
 plotMA(MAq, 1)  
 plotMA(MAq, 2)  
 plotMA(MAq, 3)  
 fit <- lmFit(MAq)  
 fit <- eBayes(fit)  
 topTable(fit)  
#   
# decideTests(fit)  
# TestResults matrix  
# x1  
# 1 0  
# 2 0  
# 3 0  
# 4 0  
# 5 0  
# 243491 more rows ...  
# > summary(decideTests(fit))  
# x1  
# -1 115  
# 0 243206  
# 1 175  
 boxplot(MA$A)  
 par(mfrow = c(1, 1))  
 boxplot(MA$A)  
 plotDensities(RG)  
# Warning message:  
# In plotDensities.RGList(RG) : NaNs produced  
 plotDensities(MA)  
 plotDensities(MAq)  
 abline(v = 4)  
 abline(v = 4.2)