Enhancer RNAs Predict Enhancer-Gene Regulatory Links and are Critical for Enhancer Function in Neuronal systems Supplemental Code

Robert A. Phillips III & Jeremy J. Day

# Mapping Transcriptionally Active Putative Enhancers (TAPEs) to Genes

The goal of this analysis is to identify high confidence TAPE-gene pairs. To do this, transcription start sites located within 1Mb upstream or downstream from the center of the TAPE are identified. Then, pearson’s correlations are calculated using counts for the TAPEs and associated genes

## Load Libraries

First, all essential libraries are loaded for the analysis.

suppressPackageStartupMessages(library(dplyr))  
suppressPackageStartupMessages(library(Seurat))

## Warning: package 'Seurat' was built under R version 3.6.2

suppressPackageStartupMessages(library(ggplot2))  
suppressPackageStartupMessages(library(cowplot))  
suppressPackageStartupMessages(library(data.table))  
suppressPackageStartupMessages(library(VennDiagram))

## Load in Required Data

First, read in the 28,492 TAPEs identified in the pipeline.

TAPEs <- read.table(file = "~/TAPE\_RNA\_quantification\_CPKM.txt",sep = "\t",header = TRUE)

Now we are going to create a column indicating these as intergenic TAPEs. Then, we are going to create an ID column consisting of the TAPE chromosome, starting position, and ending position. Next, we will rename the first four columnns. The start and end will be referred to as five and three prime end as the SeqMonk software used during the pipeline refers to the 5’ end of all genes as the start, regardless of whehter that gene is on the - strand. Thus, referring to the start/end as five prime and three prime will allow us to correctly calculate distance later in this workflow. Finally, the center of the TAPE is calculated.

#Give a column indicating the type  
TAPEs$Type <- "Intergenic"  
  
#Create a column with a unique ID.   
TAPEs$ID <- paste(TAPEs$Chromosome,TAPEs$Start,TAPEs$End,sep = "\_")  
  
#Change the TAPEs names   
names(TAPEs)[1:4] <- c("TAPE\_ID","TAPE\_Chr","TAPE\_Five\_Prime\_End","TAPE\_Three\_Prime\_End")  
  
#Calculate the center of the TAPE   
TAPEs$TAPE\_Center <- TAPEs[,"TAPE\_Five\_Prime\_End"] + (round(abs(TAPEs[,"TAPE\_Five\_Prime\_End"] - TAPEs[,"TAPE\_Three\_Prime\_End"])/2))

Next, I will read in the genes used during TAPE identification and change the column names.

#read in the genes that were used to during enhancer identificaiton   
genes <- read.delim(file = "~/Rn6\_v95\_gtf\_genesonly\_noBS.txt",sep = "\t")  
  
#Change column naems  
names(genes) <- c("Chr","Gene\_Five\_Prime\_End","Gene\_Three\_Prime\_End","Strand","Gene")

To map the TAPEs to associated genes, we will use a for loop. As the for loop proceeds, any TAPE that maps to the gene will be input into a large list created below. Here, every element of the list corresponds to a unique gene.

# Here I make an empty list to input the annotated TAEs into  
genes\_list <- vector(mode = "list",   
 length = nrow(genes))  
#Name every element of the list a unique name for that gene. This consists of chr,5' end, 3' end, Gene name, and strand that is comma separated  
names(genes\_list) <- paste(paste(paste(paste(genes$Chr,genes$Gene\_Five\_Prime\_End,sep = ","),  
 genes$Gene\_Three\_Prime\_End,sep = ","),  
 genes$Gene,sep = ","),  
 genes$Strand,sep = ",")

Next we run a sanity check to make sure that the dataframe and list are in the same order.

#Make sure the dataframe and list are in the same order  
all(as.character(lapply(strsplit(names(genes\_list),split = ","),"[",4)) == genes$Gene) # TRUE

## [1] TRUE

all(as.character(lapply(strsplit(names(genes\_list),split = ","),"[",5)) == genes$Strand) # TRUE

## [1] TRUE

all(as.character(lapply(strsplit(names(genes\_list),split = ","),"[",1)) == genes$Chr) # TRUE

## [1] TRUE

all(as.numeric(lapply(strsplit(names(genes\_list),split = ","),"[",2)) == genes$Gene\_Five\_Prime\_End) # TRUE

## [1] TRUE

all(as.numeric(lapply(strsplit(names(genes\_list),split = ","),"[",3)) == genes$Gene\_Three\_Prime\_End) # TRUE

## [1] TRUE

# Identify TAPE-Gene Pairs

Now we annotate all genes that are 1Mbp upstream and downstream of the TAPE. This loops through the genes dataframe, and first asks if the strand of the gene is + or -. If the strand of the gene is positive all calculations are computed using the five prime end of the gene. If the strand of the gene is negative all calculations are computed using the three prime end of the gene. This search is gene-centric in that this loop identifies genes that fall within 1Mbp windows from the center of the TAPE.A progress bar will also print the progress of the loop.

Next, we identify any genes in which there were no associated TAPEs. These empty elements are then removed from the list.

#Empty vector for identification of empty list elements  
x <-vector()  
#Run loop  
for(i in 1:length(genes\_list)){  
 if(is.null(genes\_list[[i]])){  
 x <- append(x = x,i)  
 }else{  
 next  
 }  
}  
#Remove genes with no TAPEs  
genes\_list <- genes\_list[-x]

This for loop adds a column to every dataframe in the list that indicates the gene name, strand, chr, 5’end, 3’ end. The list is then unlisted to create a large dataframe that can be exported. Finally, a sanity chekc is run to make sure that no rows are duplicated.

for(i in 1:length(genes\_list)){  
 genes\_list[[i]]$Gene <- as.character(lapply(strsplit(names(genes\_list)[i],split = ","),"[",4))  
 genes\_list[[i]]$Strand <- as.character(lapply(strsplit(names(genes\_list)[i],split = ","),"[",5))  
 genes\_list[[i]]$Gene\_Chr <- as.character(lapply(strsplit(names(genes\_list)[i],split = ","),"[",1))  
 genes\_list[[i]]$Gene\_Five\_Prime\_End <- as.numeric(lapply(strsplit(names(genes\_list)[i],split = ","),"[",2))  
 genes\_list[[i]]$Gene\_Three\_Prime\_End <- as.numeric(lapply(strsplit(names(genes\_list)[i],split = ","),"[",3))  
}  
  
#unlist and make a huge dataframe  
TAPEs\_df <- rbindlist(genes\_list)   
  
#There should not be any duplicated rows, but this command is a sanity check   
TAPEs\_df <- as.data.frame(distinct(TAPEs\_df)) #433,416

Here distance is calculated. If the gene is on the + strand, the TSS is the five prime end of the gene. If the gene is on the - strand, the TSS is the three prime end of the gene.

# Calculate distance and orientation based on strand  
TAPEs\_df$Distance <- NA  
TAPEs\_df$Orientation <- NA  
TAPEs\_df$Orientation <- as.character(TAPEs\_df$Orientation)  
#Calculate distance and orientation based on strand  
#To calculate the distance, the TSS for each gene must be identified. The TSS changes for each gene's strandedness in that a + stand gene's TSS will be in the Gene\_Five\_Prime\_End column and a - strand gene's TSS will be in the Gene\_Three\_Prime End column  
TAPEs\_df$TSS <- NA  
TAPEs\_df <- TAPEs\_df %>% mutate(TSS = ifelse(Strand == "+",  
 TAPEs\_df$Gene\_Five\_Prime\_End,  
 TAPEs\_df$Gene\_Three\_Prime\_End))

Next, the orientation of the gene to the TAPE is identified and distance from center of TAPE to TSS of gene is calculated.

# #If the Gene's TSS is < the TAPE center and > TAPE-1e6 then the gene is upstream of the enhancer  
# #If the Gene's TSS is > the TAPE center and < TAPE+1e6 then the gene is downstream of the enhancer  
TAPEs\_df <- TAPEs\_df %>% mutate(Orientation = ifelse((TAPEs\_df$TSS < TAPEs\_df$TAPE\_Center) & (TAPEs\_df$TSS > (TAPEs\_df$TAPE\_Center - 1e+06)),  
 "Upstream",  
 ifelse((TAPEs\_df$TSS > TAPEs\_df$TAPE\_Center) & (TAPEs\_df$TSS < (TAPEs\_df$TAPE\_Center+1e+06)),  
 "Downstream",  
 0)  
)  
)  
#Calculate Distance   
TAPEs\_df <- TAPEs\_df %>% mutate(Distance = ifelse(Orientation == "Upstream",  
 TAPEs\_df$TAPE\_Center - TAPEs\_df$TSS,  
 TAPEs\_df$TSS - TAPEs\_df$TAPE\_Center))  
#Another sanity check that the chromosomes of the TAPEs and the genes are the same  
table(as.character(TAPEs\_df$TAPE\_Chr) == as.character(TAPEs\_df$Gene\_Chr))

##   
## TRUE   
## 433416

To calculate correlations we need TAPE and gene counts. The TAPE counts are already within the TAPEs dataframe. Here, we load in mRNA count information. Next, we keep only genes in which there is one annotation. Finally, only useful columns are kept.

#Read in the Gene Probe counts identified with Seqmonk  
Gene\_Probe\_Counts <- read.delim(file = "~/mRNA\_quantification\_CPKM.txt",sep = "\t")  
#Keep only genes in which there is one annotated gene.   
Gene\_Probe\_Counts <- Gene\_Probe\_Counts[!duplicated(as.character(Gene\_Probe\_Counts$Probe)),]  
#Pull out useful columns Probe, Sample Counts  
Gene\_Probe\_Counts <- Gene\_Probe\_Counts[,c(1,13:31)]

Within this loop, pearson’s correlations are calculated.

#Make columns for correlations  
TAPEs\_df$Cortex\_Correlation <- NA  
TAPEs\_df$Hippocampus\_Correlation <- NA  
TAPEs\_df$Striatum\_Correlation <- NA  
TAPEs\_df$Global\_Correlation <- NA  
x <- vector()  
#Now calculate correlations   
for(i in 1:nrow(TAPEs\_df)){  
 #figure out which row in the Gene\_Probe\_Counts column corresponds to the gene in the TAPEs\_df column  
 row <- which(as.character(TAPEs\_df[i,"Gene"]) == as.character(Gene\_Probe\_Counts$Probe))  
 if(length(row) >0){  
 #####Cortex####  
 #Calculate the correlation  
 TAPEs\_df[i,"Cortex\_Correlation"] <- cor(y = as.numeric(TAPEs\_df[i,c(13:18)]),x = as.numeric(Gene\_Probe\_Counts[row,c(2:7)]))  
 #####Hippocampus######  
 #Calculate the correlation  
 TAPEs\_df[i,"Hippocampus\_Correlation"] <- cor(y = as.numeric(TAPEs\_df[i,c(19:24)]),x = as.numeric(Gene\_Probe\_Counts[row,c(8:13)]))  
 #######Striatum#######  
 #Calculate the correlation   
 TAPEs\_df[i,"Striatum\_Correlation"] <- cor(y = as.numeric(TAPEs\_df[i,c(25:31)]),x = as.numeric(Gene\_Probe\_Counts[row,c(14:20)]))  
 #Gloabl Correlation  
 #calculate the correlation  
 TAPEs\_df[i,"Global\_Correlation"] <- cor(y = as.numeric(TAPEs\_df[i,13:31]),x = as.numeric(Gene\_Probe\_Counts[row,2:20]))  
 }else{  
 x <- append(x = x,i)  
 }  
}

Some of the genes have count values of 0 for every sample across all cell types. This results in a correaltion value of 0. Thus, these TAPE-gene pairs are removed, leaving us with 388,605 potential TAPE-gene pairs.

TAPEs\_df <- TAPEs\_df[!is.na(TAPEs\_df$Global\_Correlation),]  
nrow(TAPEs\_df) #388,605

## [1] 388605