library(GEOquery)

library(EnsDb.Hsapiens.v79)

library(dplyr)

library(ggplot2)

library(gridExtra)

# Read Batch-Corrected logCPM tsv file from

# https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE182121

# GEO accession code GSE182121

ccount <- read.table("/Users/andrewscott/Downloads/GSE182117\_logCPM\_batchCorrected.tsv")

# restructure dataframe for ease of use

colnames(ccount) <- ccount[1,]

ccount <- ccount[-1,]

colnames(ccount)[1] <- "EnsembleIDs"

# extract Ensemble IDs from df

ensembleIDs <- ccount[,1]

# obtain corresponding HGNC Gene Symbols

geneSymbols <- AnnotationDbi::select(EnsDb.Hsapiens.v79,

key= ensembleIDs,

columns=c("SYMBOL"),

keytype="GENEID"

)

# add gene symbols to the dataframe

ccount <- left\_join(ccount,

geneSymbols,

join\_by(EnsembleIDs == GENEID)

)

# reorder the dataframe for ease of use

ccount <- dplyr::select(ccount,

"SYMBOL",

everything()

)

# extract the salient dataframe subset

genes\_of\_interest <- c("PER3", "ARNTL", "HOXB5", "TSSK6")

ccount\_sub <- dplyr::filter(ccount,

ccount[,1] %in% genes\_of\_interest

)

# convert the logCPM values the salient subset to Z-scores

# and splice back together

# remove the "non-values columns"

df <- ccount\_sub[,-c(1:2)]

# make these values doubles

df <- mutate\_all(df, function(x) as.numeric(as.character(x)))

# transpose for z-score calculations

df = t(df)

# convert df to z scores using the scale() function

df\_z <- scale(df, scale = TRUE, center = TRUE)

# re-transpose to original form

df\_z <- t(df\_z)

# add back in the gene names

ccount\_sub\_z <- cbind(ccount\_sub[,1:2],

df\_z

)

# transpose table to make it easier to add disease condition and time information

ccount\_sub\_z <- t(ccount\_sub\_z)

# create a vector of values by searching the rownames for T2D status indicator

T2D\_status <- rep("NGT", length(ccount\_sub\_z))

T2D\_status[which(grepl("T2D",rownames(ccount\_sub\_z)))] <- "T2D"

# attach to ccount\_sub\_z df

ccount\_sub\_z <- cbind(ccount\_sub\_z, T2D\_status)

# repeat this for the participant number

participant\_number <- c(rep(NA, 2),

rep("5", 8),

rep("5", 8),

rep("6", 8),

rep("6", 8),

rep("1", 7),

rep("1", 7),

rep("2", 8),

rep("2", 8),

rep("3", 8),

rep("3", 8),

rep("4", 8),

rep("4", 8),

rep("12", 7),

rep("12", 7),

rep("7", 8),

rep("7", 8),

rep("8", 8),

rep("8", 8),

rep("9", 7),

rep("9", 7),

rep("10", 8),

rep("10", 8),

rep("11", 8),

rep("11", 8)

)

ccount\_sub\_z <- cbind(ccount\_sub\_z, participant\_number)

## add the glucose high or control condition

glucose\_condition <- c(rep(NA, 2),

rep("Control", 8),

rep("High", 8),

rep("Control", 8),

rep("High", 8),

rep("Control", 7),

rep("High", 7),

rep("Control", 8),

rep("High", 8),

rep("Control", 8),

rep("High", 8),

rep("Control", 8),

rep("High", 8),

rep("Control", 7),

rep("High", 7),

rep("Control", 8),

rep("High", 8),

rep("Control", 8),

rep("High", 8),

rep("Control", 7),

rep("High", 7),

rep("Control", 8),

rep("High", 8),

rep("Control", 8),

rep("High", 8)

)

ccount\_sub\_z <- cbind(ccount\_sub\_z, glucose\_condition)

# add biopsy time information

# NB spaces show where gaps occur in the data.

# the number after the hash is the number of times there

# are for that row/participant

biopsy\_times <- c(rep(NA, 2),

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18, 30,36,42,48,54), #7

c(12,18, 30,36,42,48,54), #7

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12, 24,30,36,42,48,54), #7

c(12,18,24,30,36,42,48 ), #7

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36, 48,54), #7

c(12, 24,30,36,42,48,54), #7

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54), #8

c(12,18,24,30,36,42,48,54) #8

)

ccount\_sub\_z <- cbind(ccount\_sub\_z, biopsy\_times)

# tidy up the matrix and convert to df

colnames(ccount\_sub\_z)[1:4] <- c("PER3", "ARNTL", "HOXB5", "TSSK6")

ccount\_sub\_z <- ccount\_sub\_z[-(1:2),]

ccount\_sub\_z <- as.data.frame(ccount\_sub\_z)

# ensure that logCPM values and times are coded as doubles

ccount\_sub\_z$PER3 <- as.numeric(ccount\_sub\_z$PER3)

ccount\_sub\_z$ARNTL <- as.numeric(ccount\_sub\_z$ARNTL)

ccount\_sub\_z$HOXB5 <- as.numeric(ccount\_sub\_z$HOXB5)

ccount\_sub\_z$TSSK6 <- as.numeric(ccount\_sub\_z$TSSK6)

ccount\_sub\_z$biopsy\_times <- as.numeric(ccount\_sub\_z$biopsy\_times)

# make a data subset with only the "Control" glucose condition

ccount\_sub\_z\_c<- dplyr::filter(ccount\_sub\_z, glucose\_condition == "Control")

# custom function to perform harmonic regression and plot

plot\_harmonic\_regression <- function(data, gene, Tau = 24) {

# filter data for the specific gene

gene\_data <- data %>% filter(dataset == gene)

# perform harmonic regression for NGT

model\_NGT <- lm(y ~ sin(2\*pi/Tau \* x) + cos(2\*pi/Tau \* x), data = gene\_data[gene\_data$z == "NGT", ])

# perform harmonic regression for T2D

model\_T2D <- lm(y ~ sin(2\*pi/Tau \* x) + cos(2\*pi/Tau \* x), data = gene\_data[gene\_data$z == "T2D", ])

# create a data frame for smooth predictions

pred\_data <- data.frame(x = seq(min(gene\_data$x, na.rm = TRUE), max(gene\_data$x, na.rm = TRUE), length.out = 100))

# predict for NGT

pred\_data$y\_NGT <- predict(model\_NGT, newdata = pred\_data)

# predict for T2D

pred\_data$y\_T2D <- predict(model\_T2D, newdata = pred\_data)

# custom plotting

p <- ggplot(gene\_data, aes(x = x, y = y, color = z)) +

geom\_point(alpha = 0.5) +

geom\_line(data = pred\_data, aes(x = x, y = y\_NGT), color = "black") +

geom\_line(data = pred\_data, aes(x = x, y = y\_T2D), color = "red") +

scale\_color\_manual(values = c("T2D" = "red", "NGT" = "black")) +

scale\_x\_continuous(breaks=seq(12,54,6)) +

theme\_bw() +

theme(strip.background = element\_rect(fill = "grey80"),

legend.position = "right") +

labs(x="Time (Hours)",

y = "Expression (logCPM Z-Scores)",

title = "") +

ylim(-3,3)

return(p)

}

# prepare data for plotting

plot\_data <- list(

transmute(ccount\_sub\_z\_c, x=biopsy\_times, y=ARNTL, z = T2D\_status, dataset="ARNTL"),

transmute(ccount\_sub\_z\_c, x=biopsy\_times, y=HOXB5, z = T2D\_status, dataset="HOXB5"),

transmute(ccount\_sub\_z\_c, x=biopsy\_times, y=PER3, z = T2D\_status, dataset="PER3"),

transmute(ccount\_sub\_z\_c, x=biopsy\_times, y=TSSK6, z = T2D\_status, dataset="TSSK6")

) %>%

bind\_rows()

# plot each gene separately

plots <- lapply(c("ARNTL", "HOXB5", "PER3", "TSSK6"), function(gene) {

plot\_harmonic\_regression(plot\_data, gene)

})

# arrange plots in a grid

grid.arrange(grobs = plots, nrow = 2, top = "Circadian Gene Expression")