Biol4386\_Nucleosome\_Occupancy

Jia Zhao

3/27/2021

## This file is to summarize the steps of data analysis I did to analyze nucleosome occupancy data (wig/bigWig format) and plot the nucleosome occupancy data and motif information along with the genomic coordinates for the binding sites of transcription factors CgPho4 and ScPho4 (He, 2017).

# Overview

* Goal
* Steps
  + Data manipulation
  + Binding motif search
  + Plot
* Conclusion
* Acknowledgment

# Goal

This project is to compare the features of binding sites of an ortholog pair CgPho4 and ScPho4 (the master transcription factor for regulating phosphate starvation response. Cg: *Candida glabrata*; Sc: *Saccharomyces cerevisiae*). We hypothesize that CgPho4 is more capable of binding at the consensus CACGTG motifs that are occluded by nucleosomes, and when there are no exposed weak motif(s) nearby to allow ScPho4 and ScPho2 to use.

For more details about the background information, check this [link.](https://github.com/Intro-Sci-Comp-UIowa/biol-4386-course-project-Jia-Zhao1998).

# Steps

Here are detailed steps I took. The goal is to plot nucleosome occupancy and binding motif against the genomic position of *S. cerevisiae*.

## Data manipulation

This project requires the nucleosome occupancy data from high phosphate (Pi) background and no Pi background, and the genomic sequence to search for motifs of Pho4

### Nucleosome occupancy data

#### WT High Pi Nucleosome Occupancy Data

* Source: [Hu 2015](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4793274/)
* Strain: SKI (derived from S288C)
* Purpose: Nucleosome occupancy data of WT No Pi from [GEO](https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM1849297) in bw (BigWig) format.
* Data processing Demultiplexing was accomplished using bcl2fastq. Alignment: bowtie v0.12.7, command-line parameters -m 1 –best. Track generation: MNase-seq BED files were rendered as bedGraphs using the BEDtools genomeCoverageBed utility; these were then RPM-adjusted. Resulting bedGraphs were converted to tracks using UCSC Genome Browser’s bedGraphToBigWig utility. Genome\_build: sacCer2. Supplementary\_files\_format\_and\_content: BigWigs are per-base pair track visualizations of the MNase-seq data after RPM adjustment.## Genomic Sequence: GCF\_000146045.2\_R64\_genomic.fna.gz.
* I used bigWigtoWig package in terminal to convert the bigWig format to Wig format and store it in the local server. Size is ~90Mb: too big to upload to github

Here is the code to load wig format data: WT high Pi Nucleosome Occupancy in bigWig format (for more information about this formate, check [UCSC website](http://genome.ucsc.edu/goldenPath/help/bigWig.html)).

# Adapted from "data/07232020-Promoter\_Nucleosome\_Occupancy.Rmd"  
# Load final\_table: nucleosome occupancy data  
WT\_HighPi\_NucOccupancy <- read.table("~/Desktop/C16 Nucleosome Occupancy/data/MNase-seq.WT.YPD.R1.wig", header= FALSE, fill = TRUE) # create 4 columns fill in blank if missing value  
  
colnames(WT\_HighPi\_NucOccupancy) <- c("chromosome","start position", "end position","score") # Edit column names  
  
head(WT\_HighPi\_NucOccupancy) # Notice this file is well organized: 3694650 obs. of 4 variables: chromosome number, start position, end position, and score.

## chromosome start position end position score  
## 1 chrI 72 76 0.177880  
## 2 chrI 76 77 0.711521  
## 3 chrI 77 78 3.913360  
## 4 chrI 78 79 5.336400  
## 5 chrI 79 80 5.514280  
## 6 chrI 80 81 5.870040

#### WT No Pi Nucleosome Occupancy Data

* Source: [Xu 2011](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3127084/)
* Strain: W303
* Purpose: Nucleosome occupancy data of WT No Pi from [GEO](https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM730535) dataset in WIG format
* Data orocessing: Alignment: Sequence reads were obtained and mapped to the **S. cerevisiae (2003) genomes** using the Illumina Genome Analyzer Pipeline. All reads uniquely mapping with two or fewer mismatches were retained and read starts were summed in sliding windows of 80 bp to create summary windows. Paired-end Data: the average insert size for WT\_NoPi\_Nucleosome is 154 bp, with standard deviation of 22.8 bp  
  use .gitignore to ignore this file because it is over 10 Mb
* I plan to analyze this data in terminal and plot in R (X axis: genomic position & gene annotation, y axis: nucleosome occupancy).

Here is the code to load wig format data: WT No Pi Nucleosome Occupancy

# Adapted from "data/07232020-Promoter\_Nucleosome\_Occupancy.Rmd"  
  
# Load final\_table: nucleosome occupancy data  
setwd('~/Desktop/C16 Nucleosome Occupancy/') # set up the working directory that includes the nucleosome occupancy data.   
  
WT\_NoPi\_NucOccupancy <- read.table("~/Desktop/C16 Nucleosome Occupancy/data/GSM730535\_WT\_NoPi\_Nucleosome.wig", header= FALSE, fill = TRUE) # create 4 columns fill in blank if missing value  
head(WT\_NoPi\_NucOccupancy) # Note there are two empty columns named V3 and V4 and the definition line we don't need

## V1 V2 V3 V4  
## 1 track type=wiggle\_0 name="WT\_NoPi\_Nucleosome.wig" visibility=full  
## 2 variableStep chrom=chr1 span=1   
## 3 30 3.9537   
## 4 31 3.9537   
## 5 32 3.9537   
## 6 33 3.9537

# Clean the table: Remove the unwanted columns; organize the to three columns: chr number; postion, nucleosome occupancy value  
WT\_NoPi\_NucOccupancy <- subset(WT\_NoPi\_NucOccupancy, select= -c(V3, V4)) # remove two empty columns  
head(WT\_NoPi\_NucOccupancy)

## V1 V2  
## 1 track type=wiggle\_0  
## 2 variableStep chrom=chr1  
## 3 30 3.9537  
## 4 31 3.9537  
## 5 32 3.9537  
## 6 33 3.9537

# Specify which lines belong to which chromosome  
which(grepl("chrom=chr2", WT\_NoPi\_NucOccupancy$V2)) # Output is 204189, which means the line 204188 is the last line assigned to chr 1. It is to check the row number when the box has chrom=chr1 through 16 in it in order to store and assign this chromosome number to related position and value.

## [1] 204189

line\_chromosomes <- c(2,204189,993490,1296475,2751654,3306100,3566096,4612851,5144729,5564307,6265831,6930796,7951015,8839689,9585994,10642596) # specify the line number that stores chr number info. Use which(grepl("chrom=chr#", WT\_NoPi\_NucOccupancy$V2)) to check!  
# Optional: Filter the table based on chromosome number. Use Python to find number of lines for chr 1-16. i.e. we have the first 204188 lines that belong to chr 1  
lines\_in\_chromosome <- c(204188, 789301, 302985, 1455179, 554446, 259996, 1046755, 531878, 419578, 701524, 664965, 1020219, 888674, 746305, 1056602, 901321) # these number specifies total number of lines that belong to a chromosome (from chr1 to chr16)  
chromosome <- c("chr 1", "chr 2", "chr 3", "chr 4","chr 5", "chr 6", "chr 7", "chr 8","chr 9", "chr 10", "chr 11", "chr 12","chr 13", "chr 14", "chr 15", "chr 16") # create a varible to store the chromosome number information  
  
WT\_NoPi\_NucOccupancy <- data.frame(rep(chromosome, lines\_in\_chromosome), WT\_NoPi\_NucOccupancy) # add a new column to specify chromosome number  
colnames(WT\_NoPi\_NucOccupancy) <- c("chromosome","position", "score") # Edit column names  
head(WT\_NoPi\_NucOccupancy) # Notice the definition line and the lines that store chromosome number information are needed to be removed

## chromosome position score  
## 1 chr 1 track type=wiggle\_0  
## 2 chr 1 variableStep chrom=chr1  
## 3 chr 1 30 3.9537  
## 4 chr 1 31 3.9537  
## 5 chr 1 32 3.9537  
## 6 chr 1 33 3.9537

# Remove 17 rows that contain "variableStep" & "chrom =#" (declaration line)or the track definition line  
WT\_NoPi\_NucOccupancy <- WT\_NoPi\_NucOccupancy[-c(1,2,204189,993490,1296475,2751654,3306100,3566096,4612851,5144729,5564307,6265831,6930796,7951015,8839689,9585994,10642596),]   
head(WT\_NoPi\_NucOccupancy) # total 11543899 obs. of 3 variables

## chromosome position score  
## 3 chr 1 30 3.9537  
## 4 chr 1 31 3.9537  
## 5 chr 1 32 3.9537  
## 6 chr 1 33 3.9537  
## 7 chr 1 34 3.9537  
## 8 chr 1 35 3.9537

Notice the final table includes chromosome number, position and the score (higher score means higher nucleosome occupancy). The position corresponds to *S. cerevisiae* genome version sacCer1.

## Binding motif search

This step is done in Python. The final output is a csv file with the motif and position information stored. Notice the Python uses zero as the start position, while R and the genomic coordinates starts with 1. Keep this in mind while doing motif search in Python (add 1)! ### Goal \* Find **“CACGTG”** and 1-bp off motifs in *S.cerevisiae* genome (three versions: sacCer1, 2 and 3) in Python and import a summary of motif positions in a csv file to R.

### Step 1: Download genomic sequence data (Optiol step if data already in the data/ folder)

* Genome version: sacCer1 (or Oct. 2003 in the Saccharomyces Genome Database or SGD (<http://www.yeastgenome.org/>)). Downloaded [URL](http://hgdownload.soe.ucsc.edu/goldenPath/sacCer1/bigZips/)
* Genome version: sacCer2 (based on June 2008 in SGD). Downloaded [URL](http://hgdownload.soe.ucsc.edu/goldenPath/sacCer2/bigZips/)
* Genome version: sacCer3 (based on April 2011 in SGD). Downloaded [URL](http://hgdownload.soe.ucsc.edu/goldenPath/sacCer3/bigZips/)

### Step 2—Convert 2bit to csv in R for futher analysis in Python

* 2bit file hard to be read in Python. Converted in R to a data frame: chromosome number + sequence as one line
  + Result: \*\*‘sacCer#.2bit’ to ’S288c\_sacCer#\_genomeseq.csv’\*\*

library(rtracklayer)

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##   
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':  
##   
## clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
## clusterExport, clusterMap, parApply, parCapply, parLapply,  
## parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':  
##   
## IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':  
##   
## anyDuplicated, append, as.data.frame, basename, cbind, colnames,  
## dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,  
## grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,  
## order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,  
## rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,  
## union, unique, unsplit, which, which.max, which.min

## Loading required package: S4Vectors

##   
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':  
##   
## expand.grid

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

#Convert 2bit sequence data to a data frame and export as csv  
sacCer1<- import.2bit(con = "/Users/zhaojia/Desktop/C16 Nucleosome Occupancy/data/sacCer1.2bit")  
sacCer1 <- data.frame(sacCer1)  
write.csv(sacCer1, file = "~/Desktop/C16 Nucleosome Occupancy/data/S288c\_sacCer1\_genomeseq.csv")  
  
sacCer2<- import.2bit(con = "/Users/zhaojia/Desktop/C16 Nucleosome Occupancy/data/sacCer2.2bit")  
sacCer2 <- data.frame(sacCer2)  
write.csv(sacCer2, file = "~/Desktop/C16 Nucleosome Occupancy/data/S288c\_sacCer2\_genomeseq.csv")  
  
sacCer3<- import.2bit(con = "/Users/zhaojia/Desktop/C16 Nucleosome Occupancy/data/sacCer3.2bit")  
sacCer3 <- data.frame(sacCer3)  
write.csv(sacCer3, file = "~/Desktop/C16 Nucleosome Occupancy/data/S288c\_sacCer3\_genomeseq.csv")

### Step 3—Python: Motif search

* Find all consensus and 1-bp-off motifs in *S. cerevisiae* for all of the three versions.
  + The final outputs from Python are three ‘Motif.csv’ files including information about **“chromosome”, “motif”, “start” and “end”** position.
  + Note: pay attention to **element[2] = index + occurrence \* len(motif)** in find\_motif(motif,seq) function, which is different from my previous code. It’s a bug I fixed to get the original start position of a motif after I remove the first detected motif.

Note: Python script about generating csv of motif and position is stores in /C16 Nucleosome Occupancy/script/MotifSearch.py. For some reason, my Rstudion can’t directly run some python codes, so motif csv files (sacCer#Motif.csv were made in Jupyter lab)

### Step 4—Import “sacCer#.csv” in R

Check first six lines of motif search results for sacCer1-3.

head(sacCer1Motif)

## chromosome motif start end  
## 1 chr1 CACGTG 354 359  
## 2 chr1 CACGTG 16197 16202  
## 3 chr1 CACGTG 29374 29379  
## 4 chr1 CACGTG 31712 31717  
## 5 chr1 CACGTG 35436 35441  
## 6 chr1 CACGTG 45403 45408

head(sacCer2Motif)

## chromosome motif start end  
## 1 chrI CACGTG 354 359  
## 2 chrI CACGTG 16197 16202  
## 3 chrI CACGTG 29374 29379  
## 4 chrI CACGTG 31712 31717  
## 5 chrI CACGTG 35436 35441  
## 6 chrI CACGTG 45404 45409

head(sacCer3Motif)

## chromosome motif start end  
## 1 chrI CACGTG 354 359  
## 2 chrI CACGTG 16196 16201  
## 3 chrI CACGTG 29373 29378  
## 4 chrI CACGTG 31711 31716  
## 5 chrI CACGTG 35435 35440  
## 6 chrI CACGTG 45403 45408

# Plot

## Load the csv with the binding sites

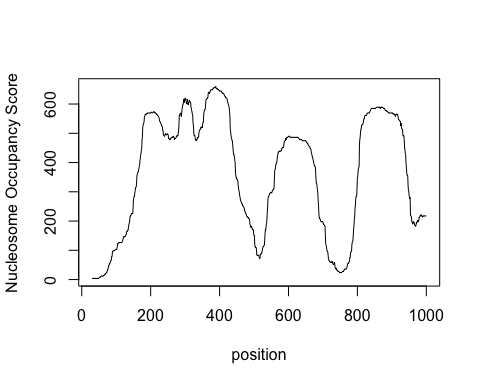
Binding\_Sites <- read.csv(file = '~/Desktop/C16 Nucleosome Occupancy/data/Binding\_Sites\_He2017.csv', sep = ",")  
head(Binding\_Sites)

## CHR START END Consensus ScPho4.bound ScPho4.occupancy.with.PHO2.No.Pi  
## 1 chr1 68474 69472 TRUE TRUE 8.682  
## 2 chr1 70668 71666 TRUE TRUE 12.532  
## 3 chr1 141266 142264 TRUE TRUE 19.526  
## 4 chr1 143368 144366 TRUE FALSE 3.103  
## 5 chr2 91616 92614 TRUE TRUE 3.968  
## 6 chr2 371065 372063 TRUE FALSE 3.143  
## ScPho4.occupancy.no.pho2.No.Pi CgPho4.bound  
## 1 8.105 TRUE  
## 2 8.645 TRUE  
## 3 8.429 TRUE  
## 4 1.945 TRUE  
## 5 1.621 TRUE  
## 6 4.106 TRUE  
## CgPho4.enrichment.with.Pho2.pho80. CgPho4.enrichment.no.pho2.pho80. GeneName  
## 1 13.610 18.049 CLN3  
## 2 7.227 7.810 CDC19  
## 3 12.117 9.139 SSA1  
## 4 4.407 8.951 SSA1  
## 5 3.785 4.733 PRS4  
## 6 5.115 4.721 NRG2  
## GeneID Peak.to.CDS ScPho4.gene.induced.with.Pho2 ScPho4.gene.induced.no.pho2  
## 1 YAL040C 1354 FALSE FALSE  
## 2 YAL038W 320 FALSE FALSE  
## 3 YAL005C 235 FALSE FALSE  
## 4 YAL005C 2337 FALSE FALSE  
## 5 YBL068W 0 FALSE FALSE  
## 6 YBR066C 766 FALSE FALSE  
## log2.ScPho4.gene.fold.with.Pho2 log2.ScPho4.gene.fold.no.pho2  
## 1 0.15 0.04  
## 2 0.39 -0.13  
## 3 1.22 -0.04  
## 4 1.22 -0.04  
## 5 1.05 -0.13  
## 6 0.08 0.14  
## CgPho4.gene.induced.with.Pho2 CgPho4.gene.induced.no.pho2  
## 1 FALSE FALSE  
## 2 FALSE FALSE  
## 3 FALSE FALSE  
## 4 FALSE FALSE  
## 5 TRUE TRUE  
## 6 TRUE TRUE  
## log2.CgPho4.gene.fold.with.Pho2 log2.CgPho4.gene.fold.no.pho2  
## 1 0.62 0.44  
## 2 0.81 0.54  
## 3 1.32 1.24  
## 4 1.32 1.24  
## 5 2.14 1.87  
## 6 1.65 1.54

# Divide the file into shared binding sites vs CgPho4-bound only sites  
Shared\_Binding\_Sites <- subset(Binding\_Sites, Binding\_Sites[, 5] == TRUE & Binding\_Sites[, 8] == TRUE)  
  
CgPho4\_Only\_Binding\_Sites <- subset(Binding\_Sites, Binding\_Sites[, 5] == FALSE & Binding\_Sites[, 8] == TRUE)

I will extract the gene names from these data frames. ## (Optional) Fun Exercise: Make a randome plot - Given the data for WT No Pi nucleosome occupancy (a data frame with three variables: chr #, position, and score); the start position, and the end position, the random\_plot will plot the nuclesome occupancy value. The default will plot chr 1: 1-1000 - **Notice the genomic cooridinate is from the raw file made by Zhou Xu that alligns with sacCer1.**

random\_plot <- function(data= WT\_NoPi\_NucOccupancy, chromosome="chr 1", start=1, end=1000){  
 # data is a data frame; chromosome is a string/character and a column of data, start and end are integers  
 sub\_data <- data[data$chromosome == chromosome,]  
 position <- as.numeric(sub\_data$position)  
 sub\_data <- sub\_data[position >= start & position <= end,]  
 with(sub\_data, plot(position, score, pch = 19, xlab = "position", ylab= "Nucleosome Occupancy Score", type = "l"))  
}  
random\_plot()

 ## Translate Genomic Coordinates: UCSC LiftOver The nucleosome occupancy data for WT high Pi and No Pi were mapped to sacCer2 and sacCer1 genome of *S. cerevisiae*, which makes them iNcomparable, so I plan to use [LiftOver](https://genome.ucsc.edu/cgi-bin/hgLiftOver) to translate coordinates to sacCer3 version, which was used by He to identify the binding sites of Pho4. - For example, sacCer 3 gene PHO5 coordinates are, while it’s chr2:431003-431523 in sacCer1. - I will do it later ## Load Gene Annotation File for sacCer1, sacCer2 and sacCer3 from UCSC The final output will be a data frame that stores information about gene name, start position, end position, strand, and chromosome number.

# UCSC: sacCer1 gene annotation: 5769 obs of 5 var (unique)  
S288cAnnotation\_sacCer1 <- read.delim(file = '~/Desktop/C16 Nucleosome Occupancy/data/sgdGene.txt', header = FALSE)  
# Information from http://hgdownload.soe.ucsc.edu/goldenPath/sacCer1/database/sgdGene.txt.gz  
colnames(S288cAnnotation\_sacCer1) <- c("gene", "chromosome", "strand", "start", "end", "cds start", "cds end", "exon counts", "exon start", "exon end", "protein ID")  
S288cAnnotation\_sacCer1 <- S288cAnnotation\_sacCer1[c(-6:-11)]  
  
# UCSC: sacCer2 (SGD June 2008) gene annotation: 6717 obs of 5 var (unique)  
S288cAnnotation\_sacCer2 <- read.delim(file = '~/Desktop/C16 Nucleosome Occupancy/data/sacCer2.txt', header = FALSE)  
# Information from http://hgdownload.soe.ucsc.edu/goldenPath/sacCer1/database/sgdGene.txt.gz  
colnames(S288cAnnotation\_sacCer2) <- c("number", "gene", "chromosome", "strand", "start", "end", "cds start", "cds end", "exon counts", "exon start", "exon end", "protein ID")  
S288cAnnotation\_sacCer2 <- S288cAnnotation\_sacCer2[c(-1,-7:-12)]  
  
# UCSC: Apr. 2011 (SacCer\_Apr2011/sacCer3) assembly of the S. cerevisiae genome (sacCer3, Saccharomyces cerevisiae S288c assembly from Saccharomyces Genome Database (GCA\_000146055.2)). 6692 obs of 5 var (unique)  
S288cAnnotation\_sacCer3 <- read.delim(file = '~/Desktop/C16 Nucleosome Occupancy/data/sacCer3.txt', header = FALSE)  
# Information from http://hgdownload.soe.ucsc.edu/goldenPath/sacCer1/database/sgdGene.txt.gz  
colnames(S288cAnnotation\_sacCer3) <- c("number", "gene", "chromosome", "strand", "start", "end", "cds start", "cds end", "exon counts", "exon start", "exon end", "protein ID")  
S288cAnnotation\_sacCer3 <- S288cAnnotation\_sacCer3[c(-1,-7:-12)]

## Nucleosome Occupancy plot of promoter region of a gene

Load plot function for ploting the nucleosome occupancy of the promoter region of a specific gene (upsteam 800 bp of CDS).

# Promoter Nucleosome Occupancy Plot of a gene  
Nucleosome\_Occupancy\_PromoterPlot <- function(data = S288cAnnotation\_sacCer1, gene = "YBR093C"){  
 line2 <- data[which(data$gene == gene),]  
 if (line2$strand == "-"){  
 random\_plot(data = WT\_NoPi\_NucOccupancy, chromosome = paste("chr", line2$chromosome), start = line2$end + 1, end = line2$end + 800)  
 title(main = gene, sub = paste("chr", line2$chromosome))  
 axis(1, at = seq(0,-800,-100))  
 }else{  
 random\_plot(chromosome = paste("chr", line2$chromosome), start = line2$start - 800, end = line2$start - 1)  
 title(main = gene, sub = paste("chr", line2$chromosome))  
 axis(1, at = seq(-800,0,100))  
 }  
Nucleosome\_Occupancy\_PromoterPlot()  
}

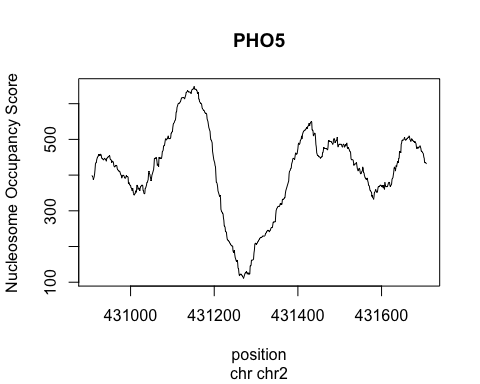
## PHO5 example: WT high Pi and No Pi plot

Derived from online UCSC LiftOver tool. - sacCer1: chr2:429504-430907 - sacCer2: chrII:429542-430945 - sacCer3: chrII:429548-430951

data <- S288cAnnotation\_sacCer1  
line2 <- data[which(data$gene == "YBR093C"),]  
line2 # find the line for this gene "YBR093C"/ PHO5 in the gene annotation file (sacCer1)

## gene chromosome strand start end  
## 3298 YBR093C chr2 - 429503 430907

# gene chromosome strand start end  
# 3298 YBR093C chr2 - 429503 430907  
  
random\_plot(data = WT\_NoPi\_NucOccupancy, chromosome = "chr 2", start = 430908, end = 431707)  
# random\_plot(data = WT\_NoPi\_NucOccupancy, chromosome = paste("chr", line2$chromosome), start = line2$end + 1, end = line2$end + 800)  
title(main = "PHO5", sub = paste("chr", line2$chromosome))  
axis(1, at = seq(0,-800,-100))



# Conclusion

# Acknowledgement

**I would like to thank my PI, Dr. Bin He, for his valuable guidance throughout this project. You provided me with the tools that I needed to choose the right direction and successfully made a progress.**