Exercise 3. Tools for integrated analysis of genome features and peaks

This exercise will work with `BEDTools`, `deepTools`, and `Homer`. We'll be mixing in some `linux` (`bash`), `awk`, and a little `R` along the way. The tools are most powerful when you can combine all of these (you could certainly swap `python` for `R`).

Part 1. BEDTools

Documentation for most (but not all!) of the BEDTools command is available here: https://bedtools.readthedocs.io/en/latest/content/overview.html#summary-of-available-tools

Or you can get help for any BEDTools command directly from the command line. The help output from BEDTools writes to `stderr`, so you can make it easier to scroll through using `less` by redirecting `stderr` like this:

```
# See all of the available BEDTools commands
bedtools -h 2>&1 | less

# Get help for the makewindows command (as
#  https://bedtools.readthedocs.io/en/latest/content/tools/makewindows.html
#  is an empty page)
bedtools makewindows -h 2>&1 | less
```

Within less, `space bar` moves you forward by one window, and the `b` key moves you back one window. When you are done, type `q` to exit less.

1. Set up your working directory for this exercise and start screen

```
# This is a comment and will not do anything if you copy/paste
#   it into the command line terminal (provided the leading #
#   is included)

##  This is my convention for the BEDTools
##  section for showing expected output
##  (i.e., prepended by ## )

mkdir -p /workdir/$USER/exercise3
cd /workdir/$USER/exercise3
cp /shared_data/epigenomics/exercise3/* /workdir/$USER/exercise3/
ls -1

##   ATAC_Rep1_peaks_filter.narrowPeak
##   ATAC_Rep2_peaks_filter.narrowPeak
##   CS_Rep1_peaks_filter.narrowPeak
##   CS_Rep2_peaks_filter.narrowPeak
##   sacCer3.chrom.sizes
```

```
# In general, don't forget to start screen (especially for commands that take a
while)
screen
```

2. Convert yeast gff3 to bed format.

Although gff3 files can be used directly as BEDTools inputs (-a "query" or -b "database") the ouput can be pretty messy because the whole gff3 file line will be included. Also, you need to make sure the chromosome names and sorting match (e.g., "chrIV" and "IV" won't match) , so this can be fixed during conversion, if need be. Here, we'll sort the chromosomes, remove the mitochondrial genome, and extract the gene names.

```
cd /workdir/$USER/exercise3

# get the gff3 annotation for the yeast sacCer3 reference genome (copy/paste the
full command)
wget "ftp://ftp.ensemblgenomes.org/pub/fungi/release-
49/gff3/saccharomyces_cerevisiae/Saccharomyces_cerevisiae.R64-1-1.49.gff3.gz"

gunzip Saccharomyces_cerevisiae.R64-1-1.49.gff3.gz
```

Then convert the gff3 file to a sorted bed with autosomal gene names. Copy/paste the entire block of code below. If it doesn't run (it's very quick -- you'll see a command prompt when it's done), then press `Enter` (or `return`) to kick it off. Whether the command runs or not will depend on if you copy/paste the final line ending.

```
awk '
BEGIN{FS=OFS="\t"}
{
    if($3=="gene" && $1!="Mito"){
        split($9,anno,";")
        split(anno[1],gene,":") # the gene name will be the 2nd element of the
array "gene"
        print $1,$4-1,$5,gene[2],".",$7
    }
}
' Saccharomyces_cerevisiae.R64-1-1.49.gff3 | \
sort -k1,1 -k2,2n > sacCer3_autosomal_genes.bed
```

Now inspect the output.

```
less sacCer3_autosomal_genes.bed
```

Within less, `space bar` moves you forward by one window, and the `b` key moves you back one window. When you are done, type `q` to exit less. Did I mention that already?

3. What genes lie directly beneath called peaks? `bedtools intersect`

In Exercise 2 from last week, you used MACS2 to call some ChIP-seq and ATAC-seq peaks, producing `*_peaks_filter.narrowPeak` files, which are in bed format. You've already copied these `*_peaks_filter.narrowPeak` files to your working folder for today's exercise in step 1 above. Below, we'll work with the ChIP-seq replicate 1 file `CS_Rep1_peaks_filter.narrowPeak`. Feel free to re-run the commands with the other three files. Also, feel free to inspect any intermediate output using `less`, or piping into less: `<command1> | <command2> | less`.

```
# Fix the chr names in the peak file (remove chr)
sed 's/chr//' CS_Rep1_peaks_filter.narrowPeak > CS_Rep1.narrowPeak.bed

# What genes (if any) fall directly under the peaks, and how large is the
overlap?
#   -wo = Write the original A and B entries plus the number of base pairs of
#         overlap between the two features.
bedtools intersect -wo -a CS_Rep1.narrowPeak.bed -b sacCer3_autosomal_genes.bed
\
    > CS_Rep1.narrowPeak.genes.bed

less CS_Rep1.narrowPeak.genes.bed  # the overlap size is the last column

# How many total overlaps are there (71)
wc -l CS_Rep1.narrowPeak.genes.bed

# How many distinct genes are overlapped? (60)
cut -f 14 CS_Rep1.narrowPeak.genes.bed | sort -u | wc -l

# What genes are overlapped by more than one peak?
cut -f 14 CS_Rep1.narrowPeak.genes.bed | sort | uniq -c | sort -nr | awk '$1>1'

##       3 YLR162W
##       3 YHR056C
##       3 YHR054C
##       2 YLR467W
##       2 YLR162W-A
##       2 YLL066W-B
##       2 YIR019C
##       2 YCR108C
```

4. What is the closest gene to each peak? `bedtools closest`

Note that `bedtools closest` requires presorted inputs.

What is the closest gene **in either direction** to each peak?

```
# -D b = report upstream or downstream distance relative to the strand of
"database" b (gene)
bedtools closest -D b -a CS_Rep1.narrowPeak.bed -b sacCer3_autosomal_genes.bed >
CS_Rep1.narrowPeak.closestGene.bed

# how many input query peaks? (328)
```

```
wc -l CS_Rep1.narrowPeak.bed

# confirm that the peak names are all distinct (I'm just paranoid...)
cut -f 4 CS_Rep1.narrowPeak.bed | sort -u | wc -l   # yes, 328

# How many peak/gene combinations? (337) # Why is this larger than 328?
wc -l CS_Rep1.narrowPeak.closestGene.bed

# How many distinct genes? (289) # Why is this smaller than 328?
cut -f 14 CS_Rep1.narrowPeak.closestGene.bed | sort -u | wc -l
```

What is the closest gene **downstream** of each peak, including overlaps?

```
#  -D b = report distance relative to the strand of "database" -b (in this case,
the gene)
#  -id = ignore queries downstream of a gene
#      Note: documentation for bedtools closest -id and -D options is confusing
(wrong???)
bedtools closest -id -D b -a CS_Rep1.narrowPeak.bed -b
sacCer3_autosomal_genes.bed > CS_Rep1.narrowPeak.closestDownStreamGene.bed

# How many peak/gene combos? (336)
wc -l  CS_Rep1.narrowPeak.closestDownStreamGene.bed

# How many distinct query peaks in the output? (328, as there should be)
cut -f 4 CS_Rep1.narrowPeak.closestDownStreamGene.bed | sort -u | wc -l

# Why are some peaks reported twice?
cut -f 4 CS_Rep1.narrowPeak.closestDownStreamGene.bed | sort | uniq -c | sort -
nr | awk '$1>1'
##      3 CS_Rep1_peak_292
##      2 CS_Rep1_peak_36
##      2 CS_Rep1_peak_200b
##      2 CS_Rep1_peak_200a
##      2 CS_Rep1_peak_149
##      2 CS_Rep1_peak_115
##      2 CS_Rep1_peak_107

# Because many genes overlap in the yeast genome, and the twice reported peaks
fall within an overlap:
cut -f 4 CS_Rep1.narrowPeak.closestDownStreamGene.bed | sort | uniq -c \
    | sort -nr | awk '$1>1{print $2}' \
    | xargs -I % grep % CS_Rep1.narrowPeak.closestDownStreamGene.bed \
    | cut -f 1-4,11-14,16,17

##  XV    720250  720472  CS_Rep1_peak_292   XV    720180  720510  YOR199W   +
0
##  XV    720250  720472  CS_Rep1_peak_292   XV    720416  720815  YOR200W   +
0
##  XV    720250  720472  CS_Rep1_peak_292   XV    720469  721708  YOR201C   -
0
##  IV    167510  167719  CS_Rep1_peak_36    IV    167358  167715  YDL162C   -
0
```

```
##  IV    167510  167719  CS_Rep1_peak_36    IV    167713  169078  YDL161W   +
0
##  XII   489913  490429  CS_Rep1_peak_200b  XII   489572  489929  YLR162W   +
0
##  XII   489913  490429  CS_Rep1_peak_200b  XII   490405  490594  YLR162W-A +
0
##  XII   489913  490429  CS_Rep1_peak_200a  XII   489572  489929  YLR162W   +
0
##  XII   489913  490429  CS_Rep1_peak_200a  XII   490405  490594  YLR162W-A +
0
##  VIII  214698  215016  CS_Rep1_peak_149   VIII  214507  214702  YHR054W-A +
0
##  VIII  214698  215016  CS_Rep1_peak_149   VIII  214532  214718  YHR055C   -
0
##  VII   123384  123597  CS_Rep1_peak_115   VII   123571  124042  YGL199C   -
0
##  VII   123384  123597  CS_Rep1_peak_115   VII   123590  124298  YGL198W   +
0
##  VI    106367  106606  CS_Rep1_peak_107   VI    106414  106732  YFL015W-A +
0
##  VI    106367  106606  CS_Rep1_peak_107   VI    106468  106963  YFL015C   -
0

# Distribution of peak distance to closest downstream gene
cut -f 17 CS_Rep1.narrowPeak.closestDownStreamGene.bed | sort -n | uniq -c |
less
```

What is the closest gene **downstream** of each peak, **excluding** overlaps?

```
# What is the closest, downstream, non-overlapping (DSNO) gene to each peak?
#     -io = ignore overlaps
bedtools closest -io -id -D b -a CS_Rep1.narrowPeak.bed -b
sacCer3_autosomal_genes.bed \
    > CS_Rep1.narrowPeak.closestDSNOGene.bed

# Distribution of peak distance to closest DSNO gene
cut -f 17 CS_Rep1.narrowPeak.closestDSNOGene.bed | sort -n | uniq -c | less
```

Once you have a list of genes either underlying or closest to your peaks, you can do additional analyses such as tests for enrichment of functional categories. This was covered in our previous RNA-seq worshop so we won't cover it again here.

5. Analysis of overlapping genes: `bedtools intersect`, `merge`, and `jaccard`

How many genes in the yeast autosome overlap?

```
bedtools intersect -wo -a sacCer3_autosomal_genes.bed -b
sacCer3_autosomal_genes.bed \
    | awk '$4!=$10' > sacCer3_overlap_genes.bed

# Count distinct number of autosomal genes that overlap (1224)
cut -f 4 sacCer3_overlap_genes.bed | sort -u | wc -l
```

What proportion of the autosomal "gene space" consists of overlapping genes?

```
# Get the "autosomal gene space" (prop. of genome falling within a gene) by
merging all overlaps
bedtools merge -c 4 -o collapse -i sacCer3_autosomal_genes.bed >
sacCer3_geneSpace.bed

# Get the distinct overlaps
bedtools merge -c 4 -o distinct -i sacCer3_overlap_genes.bed >
sacCer3_geneOverlapSpace.bed

# What proportion of the autosomal gene space consists of overlapping genes?
(11.8%)
bedtools jaccard -a sacCer3_geneOverlapSpace.bed -b sacCer3_geneSpace.bed

##  intersection  union    jaccard    n_intersections
##  1015525       8636593  0.117584   574
```

6. Similarity between replicates and ChIP-seq vs ATAC-seq: `bedtools jaccard`

Below is kind of a "toy" example using our ChIP-seq and ATAC-seq peak calls. It is based on a more full-blown example using Dnase hypersensivity data, that you can read about here: http://quinlanlab.org/tutorials/bedtools/bedtools.html#a-jaccard-statistic-for-all-400-pairwise-comparisons

Calculate the Jaccard similarity coefficient (Jaccard index) for all pairwise combinations of `*_peaks_filter.narrowPeak` files using gnu parallel:

```
parallel "bedtools jaccard -a {1}_peaks_filter.narrowPeak -b
{2}_peaks_filter.narrowPeak \
    | awk 'NR>1' \
    | cut -f 3 \
    > {1}.{2}.jaccard" \
    ::: $(ls *_peaks_filter.narrowPeak | sed 's/_peaks_filter.narrowPeak//') \
    ::: $(ls *_peaks_filter.narrowPeak | sed 's/_peaks_filter.narrowPeak//')

# see what was produced
#   ls -ltr sorts the files by modification time, with the most recently modified
at the bottom
ls -ltr

# or, more specifically
ls -l *.jaccard
```

Merge the pairwise similarities into one file (one similarity per line):

```
for jacc in *.jaccard; do
    name1=$(echo $jacc | cut -d "." -f1)
    name2=$(echo $jacc | cut -d "." -f2)
    simil=$(cat $jacc)
    echo $name1$'\t'$name2$'\t'$simil
done > pairwise.jacc.txt
```

Create a matrix that can be read into R:

```
awk '
{
    name1 = $1
    if (!(name1 in names)) {
        numNames++
        names[name1] = 1
        orderedNames[numNames] = name1
    }
    name2 = $2
    sims[name1":"name2] = ($3==1)? "1.0" : $3
}
END {
    for (n=1; n<=numNames; n++) {
        header = header "\t" orderedNames[n]
    }
    print header
    for (n1=1; n1<=numNames; n1++) {
        line = orderedNames[n1]
        for (n2=1; n2<=numNames; n2++) {
            line=line "\t" sims[orderedNames[n1]":"orderedNames[n2]]
        }
        print line
    }
}
```

```
' pairwise.jacc.txt > jacc.matrix.txt
```

Make a list of group labels that can be used in R:

```
tail -n +2 jacc.matrix.txt | cut -f 1 | cut -f 1 -d "_" > labels.txt

# Then start R
R
```

Make a PCA plot and a hierarchically clustered similarity heat map in R:

```
library(ggplot2)
x <- read.table('jacc.matrix.txt')
labels <- read.table('labels.txt')
labels[,1]
df_pca <- prcomp(x)
df_out <- as.data.frame(df_pca$x)
df_out$group <- labels[,1]
df_out
percentage <- round(df_pca$sdev / sum(df_pca$sdev) * 100, 2)
percentage <- paste( colnames(df_out), "(", paste( as.character(percentage),
"%", ")", sep="") )

p<-ggplot(df_out,aes(x=PC1,y=PC2,color=group))
p<-p+geom_point(size=3) + xlab(percentage[1]) + ylab(percentage[2])
pdf('CS_ATAC_pca.pdf')
print(p)
dev.off()

library(gplots)
library(RColorBrewer)
pdf('CS_ATAC_simClust.pdf')
heatmap.2(as.matrix(x),
          col=brewer.pal(9,"Blues"),
          margins = c(14, 14),
          density.info = "none",
          lhei = c(2, 8),
          trace="none")
dev.off()

# It's done when you see this:
## null device
##           1
```

Quit R when it's done:

```
q()
## Save workspace image? [y/n/c]: n
```

Then download `CS_ATAC_pca.pdf` and `CS_ATAC_simClust.pdf` with FileZilla and take a look. Which assay appears to be more reproducible?

7. How many genes occur in each 50kb interval ("window") of the yeast genome?

`bedtools makewindows`, `bedtools coverage`

```
# There no online documentation for makewindows. Here is how you can read the
help:
bedtools makewindows -h 2>&1 | less

# Make the windows
bedtools makewindows -g sacCer3.chrom.sizes -w 50000 \
    | sed 's/^chr//' \
    > sacCer3.chrom.50Kwin.bed

# Get the gene coverage for each window
bedtools coverage -a sacCer3.chrom.50Kwin.bed -b sacCer3_autosomal_genes.bed \
    > sacCer3.numGenes.50Kwin.bed


# Just for (nerdy) fun, let's combine it into one command
bedtools makewindows -g sacCer3.chrom.sizes -w 50000 \
    | sed 's/^chr//' \
    | bedtools coverage -a - -b sacCer3_autosomal_genes.bed \
    > sacCer3.numGenes.50KwinV2.bed

# and confirm that the output is the same (if you're paranoid, like me...)
diff -s sacCer3.numGenes.50Kwin.bed sacCer3.numGenes.50KwinV2.bed
## Files sacCer3.numGenes.50Kwin.bed and sacCer3.numGenes.50KwinV2.bed are
identical

# Start R
R
```

Make a plot of the coverage in R:

```
geneNum=read.delim("sacCer3.numGenes.50Kwin.bed",header=FALSE)
names(geneNum)=c("chr","start","end","numGenes","bpGenic","bpInterval","propGeni
c")
geneNum$cumulBinNum=1:dim(geneNum)[1]
geneNum$color='black'
myColor='gray30'
for (myChr in unique(geneNum$chr)) {
    geneNum[geneNum$chr==myChr,]$color = myColor
    if (myColor=='gray30') {myColor='gray60'}
    else {myColor='gray30'}
}
myPlotTitle='Yeast: number of genes per 50Kb bin'
png(file='yeastNumGenesHist.png',
title=myPlotTitle,height=8.5,width=11,units="in",res=600)
```

```
plot(geneNum$cumulBinNum,geneNum$numGenes,type='h',col=geneNum$color,lend='squar
e',lwd=3.5,
  xaxs="i",xlab="Cumulative Genomic Position (x50Kb)",ylab="Number of
Genes",main=myPlotTitle)
dev.off()
```

Quit R as before, then download the `yeastNumGenesHist.png` file with FileZilla and have a look.

8. Read more about BEDTools

https://bedtools.readthedocs.io/en/latest/index.html

https://bedtools.readthedocs.io/en/latest/content/overview.html#summary-of-available-tools

http://quinlanlab.org/tutorials/bedtools/bedtools.html

https://bedtools.readthedocs.io/en/latest/content/advanced-usage.html

Quinlan AR (2014) *Curr Protoc Bioinform* https://doi.org/10.1002/0471250953.bi1112s47

https://github.com/arq5x/bedtools-protocols/blob/master/bedtools.md

https://gettinggeneticsdone.blogspot.com/2014/03/visualize-coverage-exome-targeted-ngs-bedtools.html

bedtools-discuss Google group: https://groups.google.com/g/bedtools-discuss

Part 2. deepTools

1. Convert bam files to bigWig files.

As it is much more efficient to run deepTools on bigWig instead of bam files, you will first convert the two supplied bam files to bigWig files.

DeepTools are Python3 code. set Linux environment on BioHPC to run deepTools.

```
export PATH=/programs/deeptools-3.5/bin:$PATH

export PYTHONPATH=/programs/deeptools-3.5/lib64/python3.6/site-
packages:/programs/deeptools-3.5/lib/python3.6/site-packages
```

Run "bamCoverage" to convert bam files to bigWig files. Recommended setting for MNase-seq data:

--binsize 1: set bin size to 1 nucleotide, which is the highest resolution.

--MNase: set MNase mode, so that only central 3 nucleotide is used.

```
bamCoverage --numberOfProcessors 4  --bam SRR922443_s3.bam -o SRR922443_s3.bw \
--binSize 1 \
--MNase \
```

```
--normalizeUsing CPM \
--ignoreForNormalization Mito \
--minFragmentLength 100 \
--maxFragmentLength 300

bamCoverage --numberOfProcessors 4  --bam SRR922443_s4.bam -o SRR922443_s4.bw \
--binSize 1 \
--MNase \
--normalizeUsing CPM \
--ignoreForNormalization Mito \
--minFragmentLength 100 \
--maxFragmentLength 300
```

After this, you should see two bigWig files: SRR922443_s3.bw and SRR922443_s4.bw

2. Run multiBigwigSummary to get average scores in genomic regions.

The multiBigwigSummary function calculate average scores in each genomic regions for every samples. The genomic regions could be defined as sliding windows across the genome, or defined in a bed file. The output file from multiBigwigSummary can be used to plot PCA or sample correlation heatmap.

First, you will run "multiBigwigSummary bins" to calculate score in sliding windows (window size 10kb).

```
multiBigwigSummary bins --binSize 10000 --bwfiles SRR922443_s3.bw SRR922443_s4.bw
--outFileName bin_score.npz --outRawCounts bin_score.txt --numberOfProcessors 4

less bin_score.txt
```

In the output file bin_score.txt, you can see average scores in each sliding windows for the two samples. The score is the normalized coverage depth from the bigWig file, averaged by all sites in the region.

Next, you will run calculate score in each gene regions as defined in a bed file.

You are provided with a gene annotation file in gtf format. Convert the gtf file to a transcript bed file. A 6-column bed file format is used, the 6th column is "strand".

```
awk -v OFS='\t' '{if (($3=="transcript") && ($1 !="Mito")) {print $1,($4-
1),$5,".",".",$7} }' R64.gtf > transcript.bed
```

Run multiBigwigSummary:

```
multiBigwigSummary BED-file \
 --bwfiles SRR922443_s3.bw SRR922443_s4.bw  \
 --BED transcript.bed \
 -out transcript_score.npz --outRawCounts transcript_score.txt

 less transcript_score.txt
```

3. Process MNase-seq data and examine nucleosome occupancy level near the transcription start site (TSS).

Run computeMatrix on the bigWig file you created in the previous step, using the reference-point mode. The reference-points are defined in the transcript.bed file. The computeMatrix would automatically use either the start or end position of the transcript depending on the strand of the gene.

```
computeMatrix reference-point -S SRR922443_s3.bw  -R transcript.bed -a 2000 -b
2000 -o tss_ref.gz --numberOfProcessors 4 --outFileNameMatrix tss_ref.txt --
missingDataAsZero

plotProfile  -m tss_ref.gz -out SRR922443_profile.png

plotHeatmap -m tss_ref.gz -out SRR922443_heatmap.png
```

Using FileZilla to download the two .png files to your laptop and open the two files.

4.Run computeMatrix in scaled-regions mode.

You will do a small research project here. The hypothesis is that GC content is low around the transcription start and end sites of each gene. You will calculate the GC content of gene body to confirm this.

First, calculate GC% in 20bp sliding windows across the genome.

```
bedtools makewindows -g R64.genome.txt -w 20 -s 1 > R64_20bp.bed

bedtools nuc -fi R64.fa -bed R64_20bp.bed > R64.GC.txt
```

Create a bedGraph file from the GC% results. In this bedGraph file, you will use the mid-point of the 20bp window to represent the window.

```
awk  -v OFS='\t' '{print $1,$2+10,$2+11,$5}' R64.GC.txt > R64.GC.bedgraph
```

Convert the bedGraph file to bigWig file. To make this to work, you will have to extend the chromosome length by 10 bp in the chromosomal size file, due to the reason that you use the mid-point of the 20bp window in the bedGraph file.

```
export PATH=/programs/kentUtils/bin:$PATH

awk  -v OFS='\t' '{print $1,$2+10}' R64.genome.txt > R64.genome2.txt

bedGraphToBigWig R64.GC.bedgraph R64.genome2.txt R64.GC.bw
```

Run computMatrix in scale-regions mode. Use the TSS and TES as the anchor points for scaling. Also, you will include 100bp upstream and downstream of each gene.

```
computeMatrix scale-regions -S R64.GC.bw -R transcript.bed --regionBodyLength
1000 --upstream 100 --downstream 100 --startLabel TSS --endLabel TES -o gc.gz --
numberOfProcessors 4

plotProfile -m gc.gz -out gc_profile.png

plotHeatmap -m gc.gz -out gc_heatmap.png
```

Download the two png files and open them in your laptop.