Processing and analyzing stallion PE ChIP-Seq data

Equine FAANG Group

University of Nebraska-Lincoln (UNL)

Alexa Barber, Nicole Kingsley, Ted Kalbfleisch, Carrie Finno, Rebecca Bellone, and Jessica Petersen

Notes:

I. This workflow has been adapted from the analysis pipeline by Dr. Nicole Kingsley which is available at https://faang.org/ebi/ftp.ebi.ac.uk/faang/ftp/protocols/analyses/UCD_SOP_processing_and_analyzing_equine_PE_ChIP_data_20201230.pdf.

II. All sample fastq files were named with the following format:

```
${MARK}_${TISSUE}_${REPLICATE}_R1.fq.gz
${MARK}_${TISSUE}_${REPLICATE}_R2.fq.gz
```

a. For example: H3K27me3_Adipose_AH3_R1.fq.gz

III. All input fastq files were named with the following format:

```
INPUT${TYPE}_${TISSUE}_${REPLICATE}_R1.fq.gz
INPUT${TYPE}_${TISSUE}_${REPLICATE}_R2.fq.gz
```

a. For example: InputHist_Adipose_AH3_R2.fq.gz or InputCTCF_Brain_AH4_R1.fq.gz

IV. The workflow was performed on the Holland Computing Core's Crane slurm cluster (https://hcc.unl.edu/docs/). Slurm arrays were commonly used to submit jobs for each sample at the same time. Sample files are listed within "input.txt" files

V. The reference genome used for alignment was as follows:

EquCab3.0 (EquCab3.0_Genbank_Build.nowrap.fasta from NCBI)

VI. Genome size and genome fraction were determined computationally (Step 6)

```
genome_size= 2409143234
genome_fraction=0.97
```

VII. Highlighted regions throughout the script will need to be changed to employ the pipeline with other datasets

1. Trimming

Reads were trimmed by TrimGalore version 0.6 under default parameters to remove adapters and low-quality bases (Trim Galore). TrimGalore is a wrapper for CutAdapt (version 2.9, Martin 2011).

```
#!/bin/sh
#SBATCH --time=72:00:00
#SBATCH --partition=batch
#SBATCH --mem=48Gb
#SBATCH --job-name=trim
#SBATCH -error=<working_directory>/Trim.job.%A.err
#SBATCH --output=<working_directory>/Trim.job.%A.out

module load trim_galore/0.6
module load cutadapt/2.9

cd <working_directory>

mkdir -p Trimmed_Reads

for file in `ls ./Raw_Reads/*_R1.fq.gz`
do

echo ${file}
root=`basename -s _R1.fq.gz $file`
trim_galore ${file} Raw_Reads/${root}_R2.fq.gz --paired -o Trimmed_Reads

done
```

2. Indexing Reference

The reference genome, EquCab3.0 (Kalbfleisch *et al*. 2018), was indexed using both SAMtools version 1.15 (Li *et al*. 2009) and BWA version 0.7 (Li and Durbin 2009).

```
#!/bin/sh
#SBATCH --time=04:00:00
#SBATCH --mem=16g
#SBATCH --job-name=Index
#SBATCH --error=<working_directory>/index_job.%A.err
#SBATCH --output=<working_directory>/index_job.%A.out

module load samtools/1.15
module load bwa/0.7

cd <working_directory>

samtools faidx EquCab3.0_Genbank_Build.nowrap.fasta > \
EquCab3.0_Genbank_Build.nowrap.fasta.fai

bwa index -p EquCab3.0_Genbank_Build.nowrap.fasta -a
EquCab3.0_Genbank_Build.nowrap.fasta
```

3. Aligning Reads

Trimmed reads were then mapped to the equine reference genome, EquCab3.0 (Kalbfleisch *et al*. 2018) from NCBI using BWA-MEM version 0.7 (Li and Durbin 2009) and converted directly into bam files using SAMtools version 1.15 (Li *et al*. 2009). Alignment was completed using the slurm-genotyping package by ESRice (https://github.com/esrice/slurm-genotyping) with minor modifications.

```
### Download slurm-genotyping from github

git clone https://github.com/esrice/slurm-genotyping.git

### Move trimmed reads to ./slurm-genotyping/reads/

mv ./Trimmed_Reads/* ./slurm-genotyping/reads/

### Fill out the highlighted information in config.yaml file following the
instructions/examples within the file. Add lines for name, reads, r1, and r2
for each sample being mapped.

### Excerpt:
#################################################################
# Part 1: files
#
#
#
# This part of the config file specifies what files you're
#
# running the pipeline on. You will definitely need to
#
# edit this section.
#
#################################################################

# this is the path to the basename of the bwa indices for your reference
genome,
# i.e., just the path to the fasta file without the bwa extensions like .bwt
bwa_ref: path_to/EquCab3.0_Genbank_Build.nowrap.fasta
# this is the path to a faidx-indexed copy of your reference genome
faidx_ref: path_to/EquCab3.0_Genbank_Build.nowrap.fasta.fai

# this is a list of individuals you're genotyping. Each individual has a name
# (some ID string unique to that individual) and a list of fastq files. The
# fastq files can be gzipped or uncompressed. Individuals can have multiple
# lanes or libraries, paired or unpaired (see below for examples)
individuals:
    - name: CTCF_Brain_AH3
      reads:
          - r1: CTCF_Brain_AH3_R1_val_1.fq.gz
            r2: CTCF_Brain_AH3_R2_val_2.fq.gz
    - name: H3K4me1_Brain_AH4
      reads:
          - r1: H3K4me1_Heart_AH4_R1_val_1.fq.gz
            r2: H3K4me1_Heart_AH4_R2_val_2.fq.gz
    - name: InputCTCF_Brain_AH3
      reads:
          - r1: InputCTCF_Brain_AH3_R1_val_1.fq.gz
            r2: InputCTCF_Brain_AH3_R2_val_2.fq.gz
```

```
    - name: InputHist_Heart_AH4
      reads:
          - r1: InputHist_Heart_AH4_R1_val_1.fq.gz
            r2: InputHist_Heart_AH4_R2_val_2.fq.gz

### Edit ./slurm-genotyping/slurm_scripts/bwa_mem.sh to read:

#!/bin/bash
#SBATCH --error logs/bwa_mem.%a.err
#SBATCH --output logs/bwa_mem.%a.out

sample=$(head -n $SLURM_ARRAY_TASK_ID lists/alignments.tsv | cut -f1 | tail -
1)
inputs=$(head -n $SLURM_ARRAY_TASK_ID lists/alignments.tsv | cut -f2 | tail -
1)
inputs=$(echo $inputs | tr ',' ' ')
output=$(head -n $SLURM_ARRAY_TASK_ID lists/alignments.tsv | cut -f3 | tail -
1)

module load bwa/0.7 samtools/1.9

echo "threads: $threads; sample: $sample"
echo "bwa_ref: $bwa_ref"
echo "inputs: $inputs"
echo "output: $output"
bwa mem -M -t $threads -R "@RG\tID:${sample}\tSM:${sample}" $bwa_ref $inputs
| \
    samtools view -bh - | samtools sort -o $output
samtools index $output

### From within ./slurm-genotyping/ , submit alignment job using:

./genotype.py map
```

4. Mark Duplicates

PCR and optical duplicates were marked and removed using SAMtools version 1.15 (Li *et al*. 2009). SAMTools markdup is dependent on information provided by SAMtools collate and fixmate. SAMtools stats was used to examine read pairing and duplicate information prior to filtering.

```
#!/bin/sh
#SBATCH --time=04:00:00
#SBATCH --partition=batch
#SBATCH --mem=16Gb
#SBATCH --job-name=dup
#SBATCH --array=1-<sample_number>
#SBATCH -error=<working_directory>/dup.job.%A.%a.err
#SBATCH --output=<working_directory>/dup.job.%A.%a.out

module load samtools/1.15

cd <working_directory>

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input.txt | tail -n 1`
```

```
declare -a INPUTS
INPUTS=($LINE)
OUT=`basename ${INPUTS[0]} | cut -f 1,2,3 -d '_'`
OUT="${OUT}"

### input.txt should be a basic text file that lists the sample bam files
#CTCF_Brain_AH3_0.bam
#H3K4me1_Heart_AH4_0.bam
#InputCTCF_Brain_AH3_0.bam
#InputHist_Heart_AH4_0.bam

samtools collate -o ${OUT}_collate.bam ${INPUTS[0]}

samtools fixmate -m ${OUT}_collate.bam ${OUT}_fixmate.bam

samtools sort -o ${OUT}_sorted.bam ${OUT}_fixmate.bam

samtools markdup -s ${OUT}_sorted.bam ${OUT}_markdup.bam

samtools stats ${OUT}_markdup.bam > ${OUT}_dup.stats
```

5. Filtering, Sorting, and File Organization

Aligned reads were filtered using SAMtools version 1.15 (Li *et al*. 2009) to remove reads that did not map, had secondary alignments, failed platform/vendor quality tests, were identified as PCR or optical duplicates, or had lower than 30 for the alignment quality score. For more information about SAM flags, visit https://broadinstitute.github.io/picard/explain-flags.html. SAMtools stats was used to assess the number of usable reads available for peak calling. The final filtered and sorted files were moved to directories based on FDR values and peak topology (i.e. narrow marks with 0.01, narrow marks with 0.05 (called medium in this pipeline), and broad marks with 0.1)

```
#!/bin/sh
#SBATCH --time=02:00:00
#SBATCH --partition=batch
#SBATCH --mem=8Gb
#SBATCH --job-name=filter
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/filter.job.%A.%a.err
#SBATCH --output=<working_directory>/filter.job.%A.%a.out

module load samtools/1.15

cd <working_directory>

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_filter.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
OUT=`basename ${INPUTS[0]} | cut -f 1,2,3 -d '_'`
OUT="${OUT}"

### input_filter.text should list the duplicate marked bam files
#CTCF_Brain_AH3_markdup.bam
```

```
#H3K4me1_Heart_AH4_markdup.bam
#InputCTCF_Brain_AH3_markdup.bam
#InputHist_Heart_AH4_markdup.bam

samtools view -S -b -h -F 1804 -q 30 ${INPUTS[0]} > \
${OUT}.filtered.bam

samtools sort -@ 12 -o ${OUT}.filtered.sorted.bam ${OUT}.filtered.bam

# Identify number of usable reads pairs

samtools stats ${OUT}.filtered.sorted.bam > ${OUT}.filtered.sorted.bam.stats

### Organize files for each mark into directories based on peak topology and
FDR

mkdir -p ./Medium ./Narrow ./Broad ./Input

mv H3K4me1*.filtered.sorted.bam ./Medium
mv H3K4me3*.filtered.sorted.bam ./Narrow
mv H3K27ac*.filtered.sorted.bam ./Narrow
mv CTCF*.filtered.sorted.bam ./Narrow
mv H3K27me3*.filtered.sorted.bam ./Broad
mv Input*.filtered.sorted.bam ./Input
```

6. Determining Genome Size

Both of the programs for analyzing ChIP-Seq data in this study utilize genome size information in order to determine significance of enrichment for peak-calling. The size information is either a number or a fraction of mappable base-pairs of the genome and it is largely dependent on read size. First, all input histone files were merged with SAMtools version 1.15 to create one indexed bam with high coverage (Li et al. 2009). Then BEDtools version 2.27 was used to determine genome coverage of the combined input file, and more specifically, the amount of the genome that had no coverage and the total size (Quinlan and Hall 2010). These two bits of information were then used to calculate the measures of effective genome size.

```
#!/bin/sh
#SBATCH --time=12:00:00
#SBATCH --partition=batch
#SBATCH --mem=32Gb
#SBATCH --job-name=merge
#SBATCH --error=<working_directory>/merge.job.%A.err
#SBATCH --output=<working_directory>/merge.job.%A.out

module load samtools/1.15
module load bedtools/2.27

cd <working_directory>

samtools merge InputHist_ALL.bam <InputHist1.bam> <InputHist2.bam> ...
<InputHist.bam>

### Print 1st and 2nd column of the fai index file and separate the column by
tab.
awk -v OFS='\t' '{print $1,$2}' ./EquCab3.0_Genbank_Build.nowrap.fasta.fai \
>  genome.txt
```

```
### Determine the count of reads for each BP region of the genome

bedtools genomecov -d -ibam InputHist_ALL.bam > genome_coverage.txt

### Store number of locations with zero coverage and total bp size in a file.

echo "zeros" > count.txt
awk '$3==0 {count++} END {print count}' genome_coverage.txt >> count.txt
echo "whole genome" >> count.txt
wc -l genome_coverage.txt >> count.txt

### record genome size from the count.txt file
### genome mappability fraction can be calculated by subtracting zeros from
total genome size and dividing by total genome size

### The count.txt file should contain the following information:

zeros
64570325
whole genome
2409143234 genome_coverage.txt
```

7. Peak-Calling with MACS2

From the deduplicated bam files, peaks of enrichment were called with MACS2 version 2.1 (Zhang *et al*. 2008) for all marks. SICERpy version 0.1 (https://github.com/dariober/SICERpy, a wrapper for SICER from Zang *et al*. 2009) was also used to call peaks for the broad mark, H3K27me3.

7.1 Narrow Peaks: H3K27ac, H3K4me3, and CTCF*

*CTCF samples should be called with InputCTCF, not InputHist

```
#!/bin/sh
#SBATCH --time=12:00:00
#SBATCH --partition=batch
#SBATCH --mem=16Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/Narrow/peaks.job.%A.%a.err
#SBATCH --output=<working_directory>/Narrow/peaks.job.%A.%a.out

module load macs2/2.1

cd <working_directory>/Narrow/

mkdir -p Peak_Calls

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_peaks.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
OUT=`basename ${INPUTS[0]} | cut -f 1 -d '.'`
OUT="${OUT}"
```

```
TISSUE=`basename ${INPUTS[0]} | cut -f 2,3 -d '_' | cut -f 1 -d '.'`
### input_peaks.txt should list the final bam files
#CTCF_Brain_AH3.filtered.sorted.bam
#H3K4me3_Heart_AH4.filtered.sorted.bam
#H3K27ac_Testis_AH3.filtered.sorted.bam


### 7.1.1 Call peaks with MACS2

macs2 callpeak -t ${INPUTS[0]} -c \
../Input/InputHist_${TISSUE}.filtered.sorted.bam -f BAMPE -n
Peak_Calls/${OUT} -g \
<effective-genome-size_bp> -q 0.01 -B --SPMR --keep-dup all \
--extsize 550


### 7.1.2 Rename

mv Peak_Calls/${OUT}_peaks.narrowPeak Peak_Calls/${OUT}_Peaks.bed

### 7.1.3 Calculate enrichment with MACS2 using linear scale fold enrichment
(-m FE)

### Method calculates a score by comparing treatment value and control value
in every bin.

macs2 bdgcmp -t Peak_Calls/${OUT}_treat_pileup.bdg -c \
Peak_Calls/${OUT}_control_lambda.bdg -o \
Peak_Calls/${OUT}_FoldEnrichment.bdg -m FE -p 0.000001

### 7.1.4 Sort in-place (by overwriting from temp file)

sort -k1,1 -k2,2n -o Peak_Calls/${OUT}_FoldEnrichment.bdg \
Peak_Calls/${OUT}_FoldEnrichment.bdg

### 7.1.5 Calculate enrichment with MACS2 using log10 fold enrichment (-m
logLR)

### Method calculates a score by comparing treatment value and control value
in every bin.

macs2 bdgcmp -t Peak_Calls/${OUT}_treat_pileup.bdg -c \
Peak_Calls/${OUT}_control_lambda.bdg -o \
Peak_Calls/${OUT}_LogLR.bdg -m logLR -p 0.000001

### 7.1.6 Sort in-place (by overwriting from temp file)

sort -k1,1 -k2,2n -o Peak_Calls/${OUT}_LogLR.bdg Peak_Calls/${OUT}_LogLR.bdg
```

7.2 Medium Peaks: H3K4me1

```
#!/bin/sh
#SBATCH --time=12:00:00
#SBATCH --partition=batch
```

```
#SBATCH --mem=16Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/Medium/peaks.job.%A.%a.err
#SBATCH --output=<working_directory>/Medium/peaks.job.%A.%a.out

module load macs2/2.1

cd <working_directory>/Medium/

mkdir -p Peak_Calls

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_peaks.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
OUT=`basename ${INPUTS[0]} | cut -f 1 -d '.'`
OUT="${OUT}"
TISSUE=`basename ${INPUTS[0]} | cut -f 2,3 -d '_' | cut -f 1 -d '.'`

### input_peaks.txt should list the final bam files
#H3K4me1_Heart_AH4.filtered.sorted.bam

### 7.2.1 Call peaks with MACS2

macs2 callpeak -t ${INPUTS[0]} -c \
../Input/InputHist_${TISSUE}.filtered.sorted.bam -f BAMPE -n
Peak_Calls/${OUT} -g \
<effective-genome-size_bp> -q 0.05 -B --SPMR --keep-dup all \
--extsize 550


### 7.2.2 Rename

mv Peak_Calls/${OUT}_peaks.narrowPeak Peak_Calls/${OUT}_Peaks.bed

### 7.2.3 Calculate enrichment with MACS2 using linear scale fold enrichment
(-m FE)
### Method calculates a score by comparing treatment value and control value
in every bin.

macs2 bdgcmp -t Peak_Calls/${OUT}_treat_pileup.bdg -c \
Peak_Calls/${OUT}_control_lambda.bdg -o \
Peak_Calls/${OUT}_FoldEnrichment.bdg -m FE -p 0.000001

### 7.2.4 Sort in-place (by overwriting from temp file)

sort -k1,1 -k2,2n -o Peak_Calls/${OUT}_FoldEnrichment.bdg \
Peak_Calls/${OUT}_FoldEnrichment.bdg

### 7.2.5 Calculate enrichment with MACS2 using log10 fold enrichment (-m
logLR)
### Method calculates a score by comparing treatment value and control value
in every bin.

macs2 bdgcmp -t Peak_Calls/${OUT}_treat_pileup.bdg -c \
```

```
Peak_Calls/${OUT}_control_lambda.bdg -o \
Peak_Calls/${OUT}_LogLR.bdg -m logLR -p 0.000001

### 7.2.6 Sort in-place (by overwriting from temp file)

sort -k1,1 -k2,2n -o Peak_Calls/${OUT}_LogLR.bdg Peak_Calls/${OUT}_LogLR.bdg
```

7.3 Broad Peaks: H3K27me3 with MACS2

```
#!/bin/sh
#SBATCH --time=12:00:00
#SBATCH --partition=batch
#SBATCH --mem=16Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/Broad/peaks.job.%A.%a.err
#SBATCH --output=<working_directory>/Broad/peaks.job.%A.%a.out

module load macs2/2.1

cd <working_directory>/Broad/

mkdir -p Peak_Calls

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_peaks.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
OUT=`basename ${INPUTS[0]} | cut -f 1 -d '.'`
OUT="${OUT}"
TISSUE=`basename ${INPUTS[0]} | cut -f 2,3 -d '_' | cut -f 1 -d '.'`

### input_peaks.txt should list the final bam files
# H3K27me3_Adipose_AH3.filtered.sorted.bam

### 7.3.1 Call peaks with MACS2

macs2 callpeak -t ${INPUTS[0]} -c \
../Input/InputHist_${TISSUE}.filtered.sorted.bam -f BAMPE -n
Peak_Calls/${OUT} -g \
<effective-genome-size_bp> -q 0.05 --broad-cutoff 0.1 -B --SPMR \
--keep-dup all --extsize 550


### 7.3.2 Rename

mv Peak_Calls/${OUT}_peaks.broadPeak Peak_Calls/${OUT}_Peaks.bed

### 7.3.3 Calculate enrichment with MACS2 using linear scale fold enrichment
(-m FE)

### Method calculates a score by comparing treatment value and control value
in every bin.

macs2 bdgcmp -t Peak_Calls/${OUT}_treat_pileup.bdg -c \
```

```
Peak_Calls/${OUT}_control_lambda.bdg -o \
Peak_Calls/${OUT}_FoldEnrichment.bdg -m FE -p 0.000001

### 7.3.4 Sort in-place (by overwriting from temp file)

sort -k1,1 -k2,2n -o Peak_Calls/${OUT}_FoldEnrichment.bdg \
Peak_Calls/${OUT}_FoldEnrichment.bdg

### 7.3.5 Calculate enrichment with MACS2 using log10 fold enrichment (-m
logLR)

### Method calculates a score by comparing treatment value and control value
in every bin.

macs2 bdgcmp -t Peak_Calls/${OUT}_treat_pileup.bdg -c \
Peak_Calls/${OUT}_control_lambda.bdg -o \
Peak_Calls/${OUT}_LogLR.bdg -m logLR -p 0.000001

### 7.3.6 Sort in-place (by overwriting from temp file)

sort -k1,1 -k2,2n -o Peak_Calls/${OUT}_LogLR.bdg Peak_Calls/${OUT}_LogLR.bdg
```

8. Peak-Calling for H3K27me3 with SICERpy

```
#!/bin/sh
#SBATCH --time=12:00:00
#SBATCH --partition=batch
#SBATCH --mem=16Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/Broad/sicer.job.%A.%a.err
#SBATCH --output=<working_directory>/Broad/sicer.job.%A.%a.out

module load python/3.8
module load sicerpy/0.1

cd <working_directory>/Broad/

mkdir -p SICER_Peaks

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_sicer.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
OUT=`basename ${INPUTS[0]} | cut -f 1 -d '.'`
OUT="${OUT}"
TISSUE=`basename ${INPUTS[0]} | cut -f 2,3 -d '_' | cut -f 1 -d '.'`

### input_sicer.txt should list the final bam files
#H3K27me3_Adipose_AH3.filtered.sorted.bam

### When using PE reads, then second in pair needs to be discarded to prevent
double counting.
```

```
samtools view -F 128 -b ${INPUTS[0]} > SE_${OUT}.bam
samtools view -F 128 -b ../InputHist_${TISSUE}.filtered.sorted.bam \
> SE_InputHist_${TISSUE}.bam

samtools index -b SE_${OUT}.bam
samtools index -b SE_InputHist_${TISSUE}.bam

SICER.py -t SE_${OUT}.bam -c \
SE_InputHist_${TISSUE}.bam \
-g 4 -w 200 -fs 250 -gs <effective_genome_size_%> > \
./SICER_Peaks/${OUT}.sicer.bed
```

9. Validating and 10. Combining Replicate Peaks for MACS2 Peak Files

Combine the peak-calls from the biological replicates using BEDtools version 2.27 (Quinlan and Hall 2010) and a set of custom python scripts called Validate_Peaks.py and Validate_Peaks_Broad.py (listed below). For MACS2 files, enrichment information was stored in fold enrichment .bdg files while for SICER files, all peaks (significant and not significant) were stored in the peak-calls for each biological replicate. BEDtools version 2.27 was used to determine overlapping regions between significant peaks from one replicate and enriched regions from the other replicate as well as the reverse comparison. From there, these two sets of combined peaks are then concatonated and common peaks were merged using BEDtools version 2.27.

```
### This script will be used in each the Narrow, Medium, and Broad
directories with the respective input lists for each mark type.

#!/bin/sh
#SBATCH --time=02:00:00
#SBATCH --partition=batch
#SBATCH --mem=8Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/<Narrow,Medium,Broad>/ \
Peak_Calls/validate.job.%A.%a.err
#SBATCH --output=<working_directory>/<Narrow,Medium,Broad>/ \
Peak_Calls/validate.job.%A.%a.out

cd <working_directory>

module load bedtools/2.27

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_combine.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
OUT=`basename ${INPUTS[0]} | cut -f 1,2,3 -d '_'`
OUT="${OUT}"
TISSUE=`basename ${INPUTS[0]} | cut -f 1,2 -d '_'`
REPLICATE=`basename ${INPUTS[0]} | cut -f 3 -d '_'`

### input_combine.txt should list _Peaks.bed files
# H3K27ac_Brain_AH4_Peaks.bed (Narrow)
# H3K4me1_Adipose_AH3_Peaks.bed (Medium)
# H3K27me3_Heart_AH3_Peaks.bed (Broad MACS2)
```

```
### 9.1 for replicate AH3, look for significant regions overlapping
enrichment in rep AH4

if [ ${REPLICATE} = 'AH3' ]; then bedtools intersect -a ${INPUTS[0]} -b \
${TISSUE}_AH4_original_FoldEnrichment.bdg -wo -sorted > \
${OUT}_Peak_Regions_With_Replicate_FE.txt

else

### 9.2 for replicate AH4, look for significant regions overlapping
enrichment in rep AH3

bedtools intersect -a ${INPUTS[0]} -b
${TISSUE}_AH3_original_FoldEnrichment.bdg -wo -sorted > \
${OUT}_Peak_Regions_With_Replicate_FE.txt

fi

### The output has 15 columns with narrow peak format for rep 1 and bedgraph
format for rep 2 with the 15th column referring to overlapping bases

### Save Validate_Peaks.py script

import sys
from collections import defaultdict
peaks = defaultdict(list)
with open(sys.argv[1]) as f:
      input_file = sys.argv[1]
      character = input_file.split("_")[2]
      for line in f:
            parts = line.strip().split()
            peaks[(parts[0], parts[1], parts[2], parts[3], \
            parts[4])].append((float(parts[13]), int(parts[14])))

for k, v in peaks.iteritems():
      try:
            score = sum(x[0] * x[1] for x in v) / sum(x[1] for x in v)
      except ZeroDivisionError:
            score = 0
      if score >= 2:
            print("{}\t{}\t{}\t{}\t{}\t.".format(*k))

### 9.3.A Validate peaks for the narrow marks (H3K27ac, H3K4me1, H3K4me3, &
CTCF)

python ./Validate_Peaks.py \
${OUT}_Peak_Regions_With_Replicate_FE.txt > \
${OUT}_Peaks_Validated_by_Replicate.bed

### Save Validate_Peaks_Broad.py script

import sys
from collections import defaultdict

peaks = defaultdict(list)
```

```
with open(sys.argv[1]) as f:
      input_file = sys.argv[1]
      character = input_file.split("_")[2]
      for line in f:
            parts = line.strip().split()
            peaks[(parts[0], parts[1], parts[2], parts[3], \
            parts[4])].append((float(parts[12]), int(parts[13])))

for k, v in peaks.iteritems():
      try:
            score = sum(x[0] * x[1] for x in v) / sum(x[1] for x in v)
      except ZeroDivisionError:
            score = 0
      if score >= 1.5:
            print("{}\t{}\t{}\t{}\t{}\t.".format(*k))


### 9.3.B Validate peaks for the broad marks (H3K27me3 MACS2)

python ./Validate_Peaks_Broad.py \
${OUT}_Peak_Regions_With_Replicate_FE.txt > \
${OUT}_Peaks_Validated_by_Replicate.bed


### 10. Combining Replicates for MACS2 Peaks

#!/bin/sh
#SBATCH --time=02:00:00
#SBATCH --partition=batch
#SBATCH --mem=8Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/<Narrow,Medium,Broad>/ \
Peak_Calls/combine.job.%A.%a.err
#SBATCH --output=<working_directory>/<Narrow,Medium,Broad>/ \
Peak_Calls/combine.job.%A.%a.out

cd <working_directory>/<Narrow,Medium,Broad>/Peak_Calls/

mkdir -p Combined_Peaks


module load bedtools/2.27

LINE=`head -n ${SLURM_ARRAY_TASK_ID} final.input.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
TISSUE=`basename ${INPUTS[0]} | cut -f 1,2 -d '_'`

### final.input.txt will only include the Peaks_Validated_by_Replicate.bed
files from one replicate (in this case, AH3)
# H3K27me3_Testis_AH3_Peaks_Validated_by_Replicate.bed
# H3K27ac_Heart_AH3_Peaks_Validated_by_Replicate.bed
# H3K4me1_Lung_AH3_Peaks_Validated_by_Replicate.bed
```

```
### Concatenating validated peak files

cat ${INPUTS[0]} \
${TISSUE}_AH4_Peaks_Validated_by_Replicate.bed | sort -k1,1 -k2,2n > \
${TISSUE}_temp

if [ -s ${TISSUE}_temp ]; then

### Merging duplicated peaks

bedtools merge -i ${TISSUE}_temp -c 4,5 -o max > \
./Combined_Peaks/${TISSUE}_Combined_Peaks.bed

else

### If there are no merged peaks, then create an empty Combined_Peaks file

touch ${TISSUE}_Combined_Peaks.bed

fi

### Remove the temp file

rm ${TISSUE}_temp

fi
```

11. Combining Replicates for SICERpy Peaks

```
#!/bin/sh
#SBATCH --time=02:00:00
#SBATCH --partition=batch
#SBATCH --mem=8Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/Broad/SICER_Peaks/combine.job.%A.%a.err
#SBATCH --output=<working_directory>/Broad/SICER_Peaks/combine.job.%A.%a.out

cd <working_directory>/Broad/SICER_Peaks/

module load bedtools/2.27

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_sicer_combine.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
TISSUE=`basename ${INPUTS[0]} | cut -f 1,2 -d '_'`
REPLICATE=`basename ${INPUTS[0]} | cut -f 3 -d '_' | cut -f 1 -d '.'`

### input_sicer_combine.txt should contain .sicer.bed files
# H3K27me3_Adipose_AH3.sicer.bed
# H3K27me3_Lung_AH4.sicer.bed

### First, find significant in AH3 and at least enriched in AH4
```

```
if [ ${REPLICATE} = 'AH3' ]; then

bedtools intersect -a ${INPUTS[0]} -b ${TISSUE}_AH4.sicer.bed -u -header > \
${TISSUE}_AH3_Combined.bed

### This filters based on p-value from AH3

awk '$8 < 0.1' ${TISSUE}_AH3_Combined.bed > ${TISSUE}_AH3_Combined_1.bed

fi

done

### Second, find significant in AH4 and at least enriched in AH3

if [ ${REPLICATE} = 'AH4' ]; then

bedtools intersect -a ${INPUTS[0]} -b ${TISSUE}_AH3.sicer.bed -u -header > \
${TISSUE}_AH4_Combined.bed

### This filters based on p-value from AH4

awk '$8 < 0.1' ${TISSUE}_AH4_Combined.bed > ${TISSUE}_AH4_Combined_1.bed

fi

done

### Merge SICERpy Combined Files

#!/bin/sh
#SBATCH --time=02:00:00
#SBATCH --partition=batch
#SBATCH --mem=8Gb
#SBATCH --job-name=peaks
#SBATCH --array=1-#
#SBATCH --error=<working_directory>/Broad/SICER_Peaks/combine.job.%A.%a.err
#SBATCH --output=<working_directory>/Broad/SICER_Peaks/combine.job.%A.%a.out

cd <working_directory>/Broad/SICER_Peaks/

mkdir -p Combined_Peaks

module load bedtools/2.27

LINE=`head -n ${SLURM_ARRAY_TASK_ID} input_sicer_final.txt | tail -n 1`
declare -a INPUTS
INPUTS=($LINE)
TISSUE=`basename ${INPUTS[0]} | cut -f 1,2 -d '_'`


### input_sicer_final.txt should be only ${TISSUE}_AH3_Combined_1.bed files
# H3K27me3_Adipose_AH3_Combined_1.bed
# H3K27me3_Muscle_AH3_Combined_1.bed
```

```
### Concatenate and merge those two sets of files together

cat ${INPUTS[0]} ${TISSUE}_AH4_Combined_1.bed | sort -k1,1 -k2,2n - > \
${TISSUE}_temp.bed

### Merge the shared peaks and remove temp file

bedtools merge -i ${TISSUE}_temp.bed -header > \
./Combined_Peaks/${TISSUE}_Combined_1.bed

rm ${TISSUE}_temp.bed
```

12. Identifying Tissue-Unique Peaks

This script uses BEDtools version 2.27 (Quinlan and Hall 2010) to identify peaks that are found in only one tissue.

```
### repeat this step in each Combined_Peaks directory for MACS2 Combined
Peaks

#!/bin/sh
#SBATCH --time=01:00:00
#SBATCH --partition=batch
#SBATCH --mem=8Gb
#SBATCH --job-name=unique
#SBATCH --error=<working_directory>/Combined_Peaks/unique.job.%A.err
#SBATCH --output=<working_directory>/Combined_Peaks/unique.job.%A.out

cd <working_directory>/Combined_Peaks/

mkdir -p ./Unique

module load bedtools/2.27

### Loop through bed files to find the peak calls that are unique for MACS2

for file in `ls -p *Combined_Peaks.bed`
do echo ${file}
    root=`basename -s .bed ${file}`
    mark=`echo ${file} | awk -F '[_.]' '{print $1}'`
    tissue=`echo ${file} | awk -F '[_.]' '{print $2}'`
    if [ ! -f ${root}_unique.bed ]; then
        echo ${file}
        echo ${root}
        echo ${mark}
        echo ${tissue}

        ### Compare one file to all the other files

        bedtools intersect -a ${file} -b `ls -I ${file} . | grep ${mark} |
grep -v ${tissue}` -v -header > ./Unique/${root}_unique.bed

     fi
```

```
done

### Loop through bed files to find the peak calls that are unique for SICERpy
in the SICER_Peaks/Combined_Peaks/ directory

for file in `ls -p *Combined_1.bed`
do echo ${file}
    root=`basename -s .bed ${file}`
    mark=`echo ${file} | awk -F '[_.]' '{print $1}'`
    tissue=`echo ${file} | awk -F '[_.]' '{print $2}'`
    if [ ! -f ${root}_unique.bed ]; then
        echo ${file}
        echo ${root}
        echo ${mark}
        echo ${tissue}

        ### Compare one file to all the other files

        bedtools intersect -a ${file} -b `ls -I ${file} . | grep ${mark} |
grep -v ${tissue}` -v -header > ./Unique/${root}_unique.bed

     fi

done
```

References

1. Trim Galore. Available online: http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/ (Version 0.4.0).
2. Martin, M. Cutadapt Removes Adapter Sequences From High-Throughput Sequencing Reads. *EMBnet.journal* 2011, 17, 10-12.
3. Kalbfleisch, T.S.; Rice, E.S.; DePriest, M.S.; Walenz, B.P.; Hestand, M.S.; Vermeesch, J.R.; O'Connell,B.L.; Fiddes, I.T.; Vershinina, A.O.; et al. Improved reference genome for the domestic horse increases assembly contiguity and composition. *Commun. Biol.* 2018, 1, 197.
4. Wade, C.M.; Giulotto, E.; Sigurdsson, S.; Zoli, M.; Gnerre, S.; Imsland, F.; Lear, T.L.; Adelson, D.L.; Bailey, E.; Bellone, R.R.; et al. Genome Sequence, Comparative Analysis, and Population Genetics of the Domestic Horse. *Science* 2009, 326, 865-867.
5. Li, H.; Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 2009, 25, 1754-1760.
6. Li, H.; Handsaker, B.; Wysoker, A.; Fennell, T.; Ruan, J.; Homer, N.; Marth, G.; Abecasis, G.; Durbin, R.; 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 2009, 25, 2078-2079.
7. Picard Toolkit, GitHub Repository. Available online: http://broadinstitute.github.io/picard/ (Version 2.7.1).
8. Zhang, Y.; Liu, T.; Meyer, C.A.; Eeckhoute, J.; Johnson, D.S.; Berntein, B.E.; Nusbaum, C.; Myers, R.M.; Brown, M.; Wei, L.; et al. Model-based Analysis of ChIP-Seq (MACS). *Genome Biol.* 2008, 9, R137.
9. SICERpy, GitHub Repository. Available online: https://github.com/dariober/SICERpy (Accessed on 10 October 2019).
10. Zang, C.; Schones, D.E.; Zeng, C.; Cui, K.; Zhao, K.; Peng, W. A clustering approach for identification of enriched domains from histone modification ChIP-Seq data. *Bioinformatics* 2009, 25, 1952-1958.
11. Quinlan, A.R.; Hall, I.M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 2010, 26, 841-842.