Processing and analyzing equine histone ChIP-Seq

Equine FAANG Group

University of California, Davis (UCD)

N. B. Kingsley, Colin Kern, Huaijun Zhou, Jessica Petersen, Carrie Finno, and Rebecca Bellone

---

Notes:

I. This bioinformatic workflow is based largely on a more extensive analysis pipeline by Colin Kern, which can be found at https://github.com/kernco/functional-annotation.

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

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

a. For example: H3K27me3_Ovary_B.fq.gz

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

```
INPUT_${TISSUE}_${REPLICATE}.fq.gz
```

a. For example: Input_Ovary_B.fq.gz

IV. All fastq files were stored in a directory named Raw_Reads. The following additional directories were also created: Trimmed_Reads, BWA_Output (with several subdirectories), and Peak_Calls.

V. The reference genomes used for alignment were as follows:

(1) EquCab2.0 (Equus_caballus.EquCab2.dna_sm.toplevel.fa.gz from ENSEMBL Release 90) and
(2) EquCab3.0 (EquCab3.0_Genbank_Build.nowrap.fasta from NCBI)

a. Depending on which alignment was being used, the path to one of these genomes was saved to a variable called genome. For example:

```
genome=~/path/to/EquCab3.0_Genbank_Build.nowrap.fasta
```

VI. Genome size and genome fraction were determined computationally (Step 9), however, using a known measure such as golden path length is also common.

a. EquCab2.0 genome size:

```
genome_size=22269963111
genome_fraction=0.90
```

b. EquCab3.0 genome size:

```
genome_size=2409143234
genome_fraction=0.63
```

VII. All commands were run on the bash command-line. When processing numerous fastq files, one should consider utilizing a compute cluster with task array assignments to more efficiently process the data.

Steps:

1. Trimming

Reads were trimmed by TrimGalore version 0.4.0 under default parameters to remove adapters and low-quality bases (Trim Galore). TrimGalore is a wrapper for CutAdapt (version 1.8.3, Martin 2011).

```
# trim-galore/0.4.0 cutadapt/1.8.3

#1 Trim reads for quality and automatic adapter trimming

mkdir -p Trimmed_Reads

for file in `ls ./Raw_Reads`
do

echo ${file}
trim_galore Raw_Reads/${file} -o Trimmed_Reads

done
```

2. Indexing Reference and 3. Aligning Reads

Trimmed reads were then mapped to the equine reference genome, EquCab3.0 (Kalbfleisch *et al.* 2018) from NCBI (or the previous reference genome, EquCab2.0 (Wade *et al.* 2009) from ENSEMBL release 90, using BWA-MEM version 0.7.9a (Li and Durbin 2009) and converted directly into bam files using SAMtools version 1.9 (Li *et al.* 2009).

```
# bwa/0.7.9a samtools/1.9

#2 Index reference genome if needed
# Only have to do this once as long as you don't delete the files
bwa index -p ${genome} -a bwtsw ${genome}

#3 Align reads to reference genome
mkdir -p BWA_Output/Align

for file in `ls -p ./Trimmed_Reads |grep -v '/'`
do

echo ${file}
root=`basename -s _trimmed.fq.gz ${file}`

if [ ! -f BWA_Output/Align/$root.aligned.bam ]; then

echo ${file}
bwa mem -M -t 3 ${genome} Trimmed_Reads/$file | samtools view -bS - \
> BWA_Output/Align/${root}.aligned.bam

fi
```

```
done
```

4. Filtering and 5. Sorting

Aligned reads were filtered using SAMtools version 1.9 (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

Filtered reads were then sorted using SAMtools version 1.9 (Li *et al.* 2009).

```
# samtools/1.9

#4 Loop through all files in BWA_Output to filter them for quality using samtools.

for file in `ls -p ./Align | grep -v '/'`
do

echo ${file}
root=`basename -s .aligned.bam ${file}`

if [ ! -f Filter/${root}.filtered.bam ]; then

echo ${file}
samtools view -h -F 1804 -q ${quality} Align/${file} | samtools view -S -b - \
> Filter/${root}.filtered.bam

fi

done


# Make a subdirectory in BWA_Output if it doesn't already exist.
mkdir -p Sort

#5 Loop through all files in BWA_Output/Filter to sort using samtools.

for file in `ls -p ./Filter |grep -v '/'`
do

echo ${file}
root=`basename -s .filtered.bam ${file}`

if [ ! -f Sort/${root}.sorted.bam ]; then

echo ${file}
samtools sort -@ 12 -o Sort/${root}.sorted.bam Filter/${file}

fi

done
```

6. Marking Duplicates, 7. Deduplicating, and 8. Organizing Files

Picardtools version 2.7.1 was used to mark pcr and optical duplicates from the sorted bam files (http://broadinstitute.github.io/picard/). **Warning: This step is very slow and memory intensive.** Then SAMtools version 1.9 was used to remove the marked duplicates (Li *et al.* 2009). Finally, we moved the deduplicated files into 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)

```
#picardtools/2.7.1 samtools/1.9

#6 Make a subdirectory in BWA_Output if it doesn't already exist.
mkdir -p Duplicates

# Loop through all files (no subdirectories) in BWA_Output/Sort to mark PCR and
optical duplicates using picardtools.

for file in `ls -p ./Sort |grep -v '/'`
do

echo ${file}
root=`basename -s .sorted.bam ${file}`

if [ ! -f Duplicates/${root}.duplicate-marked.bam ]; then

echo "$file"
picard-tools MarkDuplicates INPUT=Sort/${file} \
OUTPUT=Duplicates/${root}.duplicate-marked.bam \
METRICS_FILE=Duplicates/${root}.dup_metrics REMOVE_DUPLICATES=false \
ASSUME_SORTED=true VALIDATION_STRINGENCY=LENIENT

fi

done


#7 Loop through all files in BWA_Output/Duplicates to remove marked dups with samtools.
for file in `ls -p ./Duplicates/*duplicate-marked.bam`
do

echo ${file}
root=`basename -s .duplicate-marked.bam ${file}`

if [ ! -f Duplicates/${root}.duplicate-removed.bam ]; then

echo ${file}
samtools view -F 1024 -b ${file} > Duplicates/${root}.duplicate-removed.bam

fi

done


#8 Organize files for each mark into directories based on peak topology and FDR
mkdir -p ./Medium ./Narrow ./Broad ./Input
cd Duplicates
```

```
# move the files for each mark into respective directory
mv H3K4me1*duplicate-removed.bam ../Medium
mv H3K4me3*duplicate-removed.bam ../Narrow
mv H3K27ac*duplicate-removed.bam ../Narrow
mv H3K27me3*duplicate-removed.bam ../Broad
mv Input*duplicate-removed.bam ../Input
```

9. 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 files were merged with SAMtools version 1.9 to create one indexed bam with high coverage (Li *et al.* 2009). Then BEDtools version 2.27.1 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.

```
# bedtools/2.27.1 samtools/1.9

#A. Merge bam files from all inputs to generate file with sufficient depth across genome.
samtools merge Input_ALL_AB.bam `echo Sort/Input*_A.sorted.bam` \
`echo Sort/Input*_B.sorted.bam`

#B. Save reference name without suffix to variable.
root=`basename -s .gz ${genome}`

#C. Set-up genome file for bedtools to use.
# Note: This is different from previous reference index.
samtools faidx ${genome} -o ${root}.fai

#D. Print 1st and 2nd column of the fai index file and separate the column by tab.
awk -v OFS='\t' '{print $1,$2}' ${root}.fai > ${root}.txt

#E. Determine the count of reads for each BP region of the genome
bedtools genomecov -d -ibam Input_ALL_AB.bam > ${root}_genome_coverage.txt

#F. 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}' ${root}_genome_coverage.txt >> count.txt
echo "whole genome" >> count.txt
wc -l ${root}_genome_coverage.txt >> count.txt
```

10. Peak-Calling

From the deduplicated bam files, peaks of enrichment were called with MACS2 version 2.1.1.20160309 (Zhang *et al.* 2008) for all marks. SICERpy version 0.1.1 (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.

```
# python/2.7.14 SICERpy/0.1.1 MACS/2.1.1.20160309

# Make a directory for peak-calls if it doesn't already exist.
mkdir -p Peak_Calls
mkdir -p Sicer_Peak_Calls
```

```
# NARROW PEAKS

#10.1 Loop through all files (no subdirectories) in BWA_Output/Narrow to do peak-calling.
for file in `ls -p BWA_Output/Narrow/`
do echo ${file}

root=`basename -s .duplicate-removed.bam ${file}`
tissue=`echo ${file} | awk -F '[_.]' '{print $2,$3}' OFS=_`

if [ ! -f Peak_Calls/${root}_Peaks.bed ]; then

echo ${file}
echo ${tissue}
echo 'Macs2 peak-calling'

#10.1A.1 Call peaks with MACS2
macs2 callpeak -t BWA_Output/Narrow/${file} -c \
BWA_Output/Input/Input_${tissue}.duplicate-removed.bam -f BAM -n Peak_Calls/${root} -g \
${genome_size} -q 0.01 -B --SPMR --keep-dup all --fix-bimodal --extsize 200

#10.1A.2 Rename
mv Peak_Calls/${root}_peaks.narrowPeak Peak_Calls/${root}_Peaks.bed

#10.1A.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/${root}_treat_pileup.bdg -c \
Peak_Calls/${root}_control_lambda.bdg -o \
Peak_Calls/${root}_FoldEnrichment.bdg -m FE -p 0.000001

#10.1A.4 Sort in-place (by overwriting from temp file)
sort -k1,1 -k2,2n -o Peak_Calls/${root}_FoldEnrichment.bdg \
Peak_Calls/${root}_FoldEnrichment.bdg

#10.1A.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/${root}_treat_pileup.bdg -c \
Peak_Calls/${root}_control_lambda.bdg -o \
Peak_Calls/${root}_LogLR.bdg -m logLR -p 0.000001

#10.1A.6 Sorti in-place (by overwriting from temp file)
sort -k1,1 -k2,2n -o Peak_Calls/${root}_LogLR.bdg Peak_Calls/${root}_LogLR.bdg

fi

done


# MEDIUM PEAKS

#10.2 Loop through all files (no subdirectories) in BWA_Output/Medium to do peak-calling
for file in `ls -p BWA_Output/Medium/`
do echo ${file}
```

```
root=`basename -s .duplicate-removed.bam ${file}`
tissue=`echo ${file} | awk -F '[_.]' '{print $2,$3}' OFS=_`

if [ ! -f Peak_Calls/${root}_Peaks.bed ]; then

echo ${file}
echo 'Macs2 peak-calling'

#10.2A.1 Call peaks with MACS2
macs2 callpeak -t BWA_Output/Medium/${file} -c \
BWA_Output/Input/Input_${tissue}.duplicate-removed.bam -f BAM -n \
Peak_Calls/${root} -g ${genome_size} -q 0.05 -B --SPMR --keep-dup all \
--fix-bimodal --extsize 200

#10.2A.2 Rename
mv Peak_Calls/${root}_peaks.narrowPeak Peak_Calls/${root}_Peaks.bed

#10.2A.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/${root}_treat_pileup.bdg -c \
Peak_Calls/${root}_control_lambda.bdg -o \
Peak_Calls/${root}_FoldEnrichment.bdg -m FE -p 0.000001

#10.2A.4 Sort in-place (by overwriting from temp file)
sort -k1,1 -k2,2n -o Peak_Calls/${root}_FoldEnrichment.bdg \
Peak_Calls/${root}_FoldEnrichment.bdg

#10.2A.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/${root}_treat_pileup.bdg -c \
Peak_Calls/${root}_control_lambda.bdg -o \
Peak_Calls/${root}_LogLR.bdg -m logLR -p 0.000001

#10.2A.6 Sort in-place (by overwriting from temp file)
sort -k1,1 -k2,2n -o Peak_Calls/${root}_LogLR.bdg Peak_Calls/${root}_LogLR.bdg

fi

done


# BROAD PEAKS with MACS2

#10.3 Loop through all files (no subdirectories) in BWA_Output/Broad to do peak-calling
for file in `ls -p BWA_Output/Broad/`
do echo ${file}

root=`basename -s .duplicate-removed.bam ${file}`
tissue=`echo ${file} | awk -F '[_.]' '{print $2,$3}' OFS=_`

if [ ! -f Peak_Calls/${root}_Peaks.bed ]; then

echo ${file}
```

```
echo 'Macs2 peak-calling'

#10.3A.1 Call peaks with MACS2
macs2 callpeak -t BWA_Output/Broad/${file} -c \
BWA_Output/Input/Input_${tissue}.duplicate-removed.bam -f BAM -n \
Peak_Calls/${root} -g ${genome_size} -q 0.05 --broad --broad-cutoff 0.1 -B \
--SPMR --keep-dup all --fix-bimodal --extsize 200

#10.3A.2 Rename
mv Peak_Calls/${root}_peaks.broadPeak Peak_Calls/${root}_Peaks.bed

#10.3A.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/${root}_treat_pileup.bdg -c \
Peak_Calls/${root}_control_lambda.bdg -o \
Peak_Calls/${root}_FoldEnrichment.bdg -m FE -p 0.000001

#10.3A.4 Sort in-place (by overwriting from temp file)
sort -k1,1 -k2,2n -o Peak_Calls/${root}_FoldEnrichment.bdg \
Peak_Calls/${root}_FoldEnrichment.bdg

#10.3A.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/${root}_treat_pileup.bdg -c \
Peak_Calls/${root}_control_lambda.bdg -o \
Peak_Calls/${root}_LogLR.bdg -m logLR -p 0.000001

#10.3A.6 Sort in-place (by overwriting from temp file)
sort -k1,1 -k2,2n -o Peak_Calls/${root}_LogLR.bdg Peak_Calls/${root}_LogLR.bdg


# BROAD PEAKS with SICERpy

#10.3B.1 Call peaks with SICERpy
sicer.py -t BWA_Output/Broad/${file} -c \
BWA_Output/Duplicates/Input_${tissue}.duplicate-removed.bam \
-g 4 -w 200 -fs 200 -gs ${genome_fraction} > Sicer_Peak_Calls/${root}.sicer.bed

fi

done
```

11. Combining Replicates

Combine the peak-calls from the biological replicates using BEDtools version 2.27.1 (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 2.27.1 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 2.27.1.

```
# bedtools/2.27.1 python/2.7.14
```

```
mkdir -p Combined_Peaks

#11.1 Loop through files in Peak_Calls/ to combine biological replicates for MACS2 peaks
for file in `ls -p Peak_Calls/*_Peaks.bed`
do echo ${file}

root=`basename -s _Peaks.bed ${file}`

tissue=`echo ${file} | awk -F '[_.]' '{print $1,$2,$3}' OFS=_`
replicate=`echo ${file} | awk -F '[_.]' '{print $4}'`
mark=`echo ${file} | awk -F '[_.]' '{print $2}'`

if [ ! -f Peak_Calls/${root}_Peaks_Validated_by_Replicate.bed ]; then

echo ${file}
echo ${root}
echo ${tissue}
echo ${replicate}
echo ${mark}

echo 'combining Macs peaks part 1'
if [ ${replicate} = 'Combined' ]; then echo ${file} "is already created"

else

#11.1.1 for replicate A, look for significant regions overlapping enrichment in rep B
if [ ${replicate} = 'A' ]; then bedtools intersect -a ${file} -b \
${tissue}_B_FoldEnrichment.bdg -wo -sorted > \
Peak_Calls/${root}_Peak_Regions_With_Replicate_FE.txt

else

#11.1.2 for replicate B, look for significant regions overlapping enrichment in rep A
bedtools intersect -a ${file} -b ${tissue}_A_FoldEnrichment.bdg -wo -sorted > \
Peak_Calls/${root}_Peak_Regions_With_Replicate_FE.txt

fi

fi

#11.2 There are a set of validate peaks external python scripts
if [ ${mark} = 'Calls/H3K27me3' ]; then

echo "broad method"

#11.2.1 for the broad mark
python ../Scripts/Validate_Peaks_Broad.py \
Peak_Calls/${root}_Peak_Regions_With_Replicate_FE.txt > \
Peak_Calls/${root}_Peaks_Validated_by_Replicate.bed

else

echo "narrow method"
```

```
#11.2.2 for the narrow marks
python ../Scripts/Validate_Peaks.py \
Peak_Calls/${root}_Peak_Regions_With_Replicate_FE.txt > \
Peak_Calls/${root}_Peaks_Validated_by_Replicate.bed

fi

fi

done

#11.3 After validation, the peaks can be merged
for file in `ls -p Peak_Calls/*A_Peaks_Validated_by_Replicate.bed`
do echo ${file}

root=`basename -s _Peaks_Validated_by_Replicate.bed ${file}`
tissue=`echo ${file} | awk -F '[_.]' '{print $1,$2,$3}' OFS=_`

if [ ! -f ${tissue}_Combined_Peaks.bed ]; then

echo ${file}
echo 'combining Macs peaks part 2'

#11.3.1 Concatonating validated peak files
cat Peak_Calls/${root}_Peaks_Validated_by_Replicate.bed \
${tissue}_B_Peaks_Validated_by_Replicate.bed | sort -k1,1 -k2,2n > \
${tissue}_temp

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

#11.3.2 Merging duplicated peaks
bedtools merge -i ${tissue}_temp -c 4,5 -o max > \
Combined_Peaks/${tissue}_Combined_Peaks.bed

else

#11.3.3 If there are no merged peaks, then create an empty Combined_Peaks file
touch Combined_Peaks/${tissue}_Combined_Peaks.bed

fi

#remove the temp file
rm ${tissue}_temp

fi

done


#11.4 Loop through files in Sicer_Peak_Calls/ to combine biological replicates peaks.

mkdir -p Combined_Peaks
```

```
#11.4.1 First, find significant in A and at least enriched in B.
for file in `ls -p *_A.*.bed`
do echo ${file}

root=`basename -s .bed ${file}`
tissue=`echo ${root} | awk -F '[_.]' '{print $1,$2}' OFS=_`

echo ${root}
echo ${tissue}

if [ ! -f ${tissue}.CombinedA.bed ]; then

file2=${tissue}_B*.bed
echo ${file2}

bedtools intersect -a ${file} -b ${file2} -u -header > ${tissue}.CombinedA.bed

# This filters based on p-value from A
awk '$8 < 0.1' "$tissue".CombinedA.bed > "$tissue"_CombinedA_1.bed

fi

done

#11.4.2 Second, find significant in B and at least enriched in A.
for file in `ls -p *_B.*.bed`
do echo ${file}

root=`basename -s .bed ${file}`
tissue=`echo ${root} | awk -F '[_.]' '{print $1,$2}' OFS=_`

echo ${root}
echo ${tissue}

if [ ! -f ${tissue}.CombinedB.bed ]; then

file2=../${tissue}_A*.bed
echo $file2

bedtools intersect -a ${file} -b ${file2} -u -header > ${tissue}.CombinedB.bed

# This filters based on p-value from B
awk '$8 < 0.1' ${tissue}.CombinedB.bed > ${tissue}_CombinedB_1.bed

fi

done

#11.4.3 Concatonate and merge those two sets of files together
for file in `ls -p *CombinedA_1.bed`
do echo ${file}

tissue=`echo ${file} | awk -F '[_.]' '{print $1,$2}' OFS=_`
```

```
echo ${tissue}

if [ ! -f ${tissue}.Combined_1.bed ]; then

file2=${tissue}_CombinedB_1.bed
echo ${file2}

# Concatonate the files together.
cat ${file} ${file2} | sort -k1,1 -k2,2n - > temp.bed

# merge the shared peaks and remove temp file
bedtools merge -i temp.bed -header > ${tissue}.Combined_1.bed
rm temp.bed

fi

done
```

Custom scripts needed for combining MACS2 replicates:

```
# Validate_Peaks.py

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))
```

```
# Validate_Peaks_Broad.py

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))
```

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 featues. *Bioinformatics* 2010, 26, 841-842.