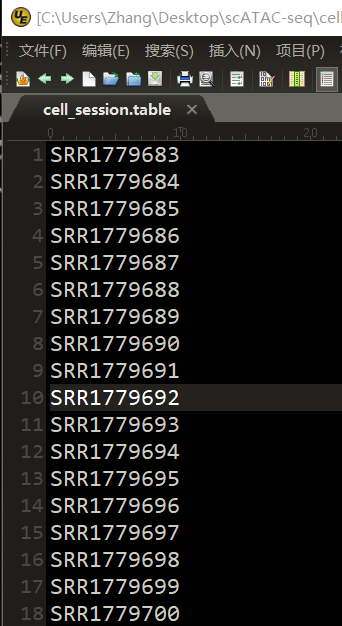
scATAC-seq pipeline

Step1: generate fragment BED file

1. Overall command:

python run\_scATAC\_pipeline.py cell\_session.table

This command will run all the scATAC-seq sample and generate the output files. cell\_session.table is a text file which contains the session number of the input sample file, see following picture. In the same folder, users should put all the fastq files, foe example, SRR1779683\_1.fastq, SRR1779683\_2.fastq.



Actually, run\_scATAC\_pipeline.py calls a shell file named order\_new.sh which contains commands for processing one scATAC-seq sample.

The final output is many files contains reads BED file, fragments BED files and kinds of QC files, all this file will be in the same folder as the input fastq files.

1. Command explanations

The following is the explanations for commands used for processing one biological sample. Users can using command “order\_new.sh SRR1779683” to run a single sample.

**# session number for a specific sample, for example sessnum=SRR1779683**

sessnum=$1

**# create folder for FASTQC and do quality control**

**# input: fastq files**

**# output: all the output file are in the "fastqc\_for\_"$sessnum folder**

mkdir "fastqc\_for\_"$sessnum

fastqc $sessnum"\_1.fastq" $sessnum"\_2.fastq" -o "fastqc\_for\_"$sessnum

**# identify adapters**

**# input: fastq files**

**# output:** **"adapters\_ident\_"$sessnum".txt"**

AdapterRemoval --identify-adapters --file1 $sessnum"\_1.fastq" --file2 $sessnum"\_2.fastq" > "adapters\_ident\_"$sessnum".txt"

**# script for extract adapter sequence**

**# input: adapters\_ident\_"$sessnum".txt**

**# output: adapters\_"$sessnum".txt**

python extract\_adapter\_using\_AdapterRemoval.py "adapters\_ident\_"$sessnum".txt" "adapters\_"$sessnum".txt"

**# remove adapters**

**# input: fastq files**

**# output: output.pair1.truncated, output.pair2.truncated and some report file**

AdapterRemoval --file1 $sessnum"\_1.fastq" --file2 $sessnum"\_2.fastq" --basename output --adapter-list "adapters\_"$sessnum".txt"

**# mapping using the output of AdapterRemoval.**

**# Input: output.pair1.truncated, output.pair2.truncated,**

**# output: $sessnum"\_mapping\_result.sam", $sessnum".obw2" and mapping report**

bowtie2 -p 6 -X2000 --dovetail --no-mixed --no-discordant --no-unal -x /home/wzhang/genome/hg19\_bowtie2/hg19 -q -1 output.pair1.truncated -q -2 output.pair2.truncated -S $sessnum"\_mapping\_result.sam" > $sessnum".obw2" 2>&1

**# convert to sorted bam**

**# input: $sessnum"\_mapping\_result.sam"**

**# output: $sessnum"\_sorted.bam"**

samtools view -b -S $sessnum"\_mapping\_result.sam" | samtools sort - > $sessnum"\_sorted.bam"

**# remove duplicates**

**# input: $sessnum"\_sorted.bam"**

**# output: $sessnum"\_sorted\_rmdup.bam", duplicates report**

java -Xmx5g -jar /home/wzhang/software/Picard/picard.jar MarkDuplicates REMOVE\_DUPLICATES=true INPUT=$sessnum"\_sorted.bam" OUTPUT=$sessnum"\_sorted\_rmdup.bam" METRICS\_FILE=picard\_rmdup.txt

**# shift reads**

**# input: $sessnum"\_sorted\_rmdup.bam"**

**# output: $sessnum"\_shifted.bed"**

bedtools bamtobed -i $sessnum"\_sorted\_rmdup.bam" | awk 'BEGIN {OFS = "\t"} ; {if ($6 == "+") print $1, $2 + 4, $3 + 4, $4, $5, $6; else print $1, $2 - 5, $3 - 5, $4, $5, $6}' - > $sessnum"\_shifted.bed"

**# remove chrM, chrY and merge reads to fragment (chrM not removed in reads file)**

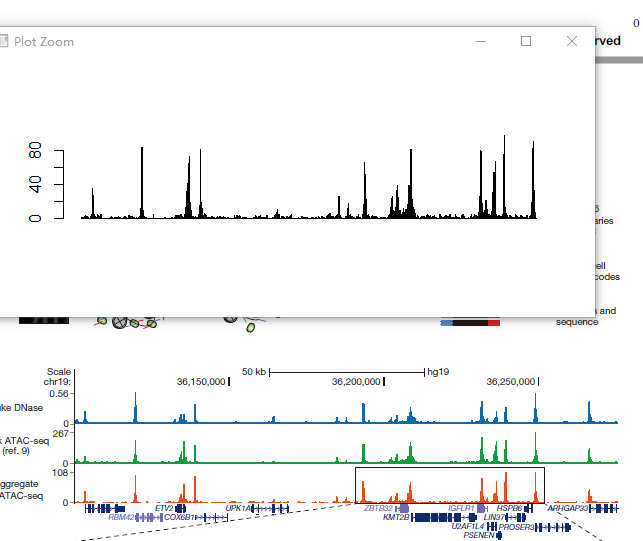
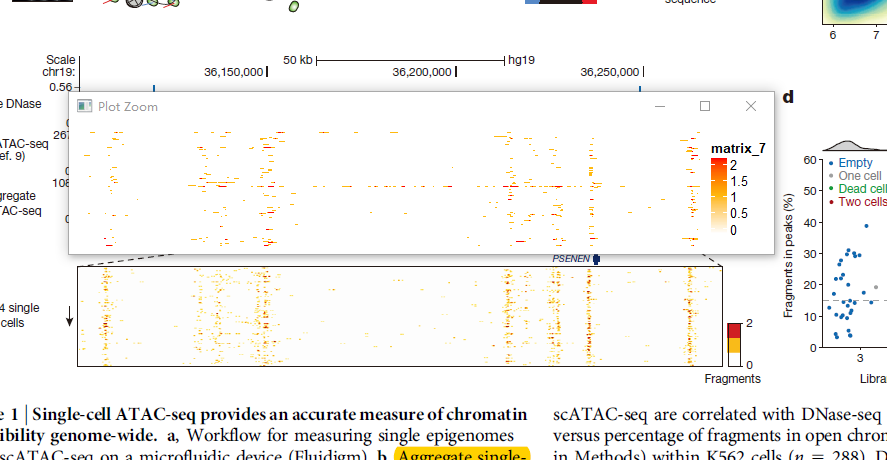
**# input: $sessnum"\_shifted.bed"**

**# output: $sessnum"\_frag.bed"**

python remove\_chrM\_and\_merge.py $sessnum"\_shifted.bed" $sessnum"\_frag.bed"

The main outputs are bamfiles, reads BED files, fragments BED files for one sample.

Some output compare are as follows.



Step2: call peaks and select

Running peaks.sh

**# merge all reads (not fragment) bed file**

**# input: all reads BED file**

**# output: reads.bed**

cat /home/wzhang/scATAC\_test/GM12878/reads\_output/\*\_shifted.bed > reads.bed

**# delete chrM and chrY**

**# input: reads.bed**

**# output: reads\_filtered.bed**

python rm\_chrM.py reads.bed reads\_filtered.bed

**# sort file**

**# input: reads\_filtered.bed**

**# output: reads\_filtered\_sorted.bed**

sortBed -i reads\_filtered.bed > reads\_filtered\_sorted.bed

**# delete tmp file**

rm -f reads.bed

rm -f reads\_filtered.bed

**# call peaks**

**# input: reads\_filtered\_sorted.bed**

**# output: output\_summits.bed, narrowpeak files**

macs2 callpeak -t reads\_filtered\_sorted.bed -f BED -g hs -n output --nomodel --nolambda --keep-dup all --call-summits

**# filter peaks in blacklist, 0.2 means overlap rate, output file: peaks\_filtered.bed**

**# input: output\_summits.bed, consensusBlacklist.bed**

**# output: peaks\_filtered.bed**

Rscript peak\_filter.R output\_summits.bed consensusBlacklist.bed peaks\_filtered.bed 0.2

**# sort by score, extract top 50000 and sort**

**# input: peaks\_filtered.bed**

**# output: top\_peaks.bed**

sort -k5rn,5 peaks\_filtered.bed | head -n 50000 - | sort -k1,1 -k2n,2 - > top\_peaks.bed