Marques et al 2021 Wt1bOE ATACSeq Preprocessing

Contents

ATACseq Preprocessing

All commands were run in bash

FASTQC

```
inputFiles_R1="./input_files_R1.txt"
array_R1=($(<$inputFiles_R1))
mkdir -p fastqc


for SAMPLE in {0..5}
do
#set input file 1 to "FL1"
FL1=${array_R1[SAMPLE]}
echo ${FL1}

fastqc ./raw_fastq/${FL1} -o ./fastqc &


done

inputFiles_R2="./input_files_R2.txt"
array_R2=($(<$inputFiles_R2))
mkdir -p fastqc


for SAMPLE in {0..5}
do
#set input file 1 to "FL1"
FL2=${array_R2[SAMPLE]}
echo ${FL2}

fastqc ./raw_fastq/${FL2} -o ./fastqc &
```

```
done

multiqc ./fastqc/* -o ./fastqc
```

Trimmomatic

```
mkdir -p trim_log
mkdir -p trim_paired
mkdir -p trim_unpaired


inputFiles_R1="./input_files_R1.txt"
inputFiles_R2="./input_files_R2.txt"

#Create the array with the links in the file
array_R1=($(<$inputFiles_R1))
array_R2=($(<$inputFiles_R2))


for SAMPLE in {0..5}
do
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%.fastq.gz}
#echo ${FL1_trimmed}

FL2=${array_R2[SAMPLE]}
#echo ${FL2}
FL2_trimmed=${FL2%.fastq.gz}
#echo ${FL2_trimmed}

#invoke java ,send FL1 and FL2 to appropriate output folders
```

```
java -jar ./Trimmomatic-0.39/trimmomatic-0.39.jar PE \
-threads 8 -phred33 -trimlog ./trim_log/${FL1}.trimlog \
./raw_fastq/${FL1} ./raw_fastq/${FL2}  \
./trim_paired/${FL1_trimmed}.pair.fastq.gz ./trim_unpaired/${FL1_trimmed}.unpair.fastq.gz \
./trim_paired/${FL2_trimmed}.pair.fastq.gz /trim_unpaired/${FL2_trimmed}.unpair.fastq.gz \
ILLUMINACLIP:/Trimmomatic-0.39/adapters/NexteraPE-PE.fa:2:30:10 LEADING:5 TRAILING:5 SLIDINGWINDOW:4:15 MINLEN:28

#add verbose option to track progress
echo "Sample ${SAMPLE} done"

done
```

Bowtie

```


inputFiles_R1="./trim_paired/input_files_R1.txt"
inputFiles_R2="./trim_paired/input_files_R2.txt"

#Create the array with the links in the file
array_R1=($(<$inputFiles_R1))
array_R2=($(<$inputFiles_R2))

export BOWTIE2_INDEXES="./Ensmbl/GRCz11/v102/bowtie_index/"
bowtie2index=bowtie_index


for SAMPLE in {0..5}
do
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%_L1*}
echo ${FL1_trimmed}

FL2=${array_R2[SAMPLE]}
```

```
#echo ${FL2}


#invoke java ,send FL1 and FL2 to appropriate output folders

echo "starting bowtie2 on ${FL1_trimmed}"

bowtie2 --threads 8 --very-sensitive -x $bowtie2index -1 ./trim_paired/${FL1} -2 ./trim_paired/${FL2} \
-S ./SAM_files/${FL1_trimmed}.sam


echo "converting ${FL1_trimmed}.sam to bam"
samtools view --threads 10 -S -b ./SAM_files/${FL1_trimmed}.sam > ./SAM_files/${FL1_trimmed}.bam


echo "converting ${FL1_trimmed}.bam to sorted.bam"
samtools sort --threads 10 -m 4G ./SAM_files/${FL1_trimmed}.bam -o ./SAM_files/${FL1_trimmed}.sorted.bam


echo "creating ${FL1_trimmed}.bam index"
samtools index -@ 10 -m 4G ./SAM_files/${FL1_trimmed}.sorted.bam ./SAM_files/${FL1_trimmed}.sorted.bam.bai


#add verbose option to track progress
echo "Sample ${FL1_trimmed} done"

done
```

Picard to get fragment lengths

```
#run picard to get fragment lengths
```

```
inputFiles_R1="./trim_paired/input_files_R1.txt"
array_R1=($(<$inputFiles_R1))
mkdir -p picard_results

for SAMPLE in {0..5}
do
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%_L1*}
echo ${FL1_trimmed}

echo "starting picard on ${FL1_trimmed}"

java -XX:ParallelGCThreads=10 -jar ./picard.jar CollectInsertSizeMetrics \
I=./SAM_files/${FL1_trimmed}.sorted.bam O=./picard_results/${FL1_trimmed}_insert_size_metrics.txt \
H=./picard_results/${FL1_trimmed}_insert_size_histogram.pdf

done
```

Remove mitochondrial chromosome and duplicates

```
#run samtools to remove mitochondria chromosome

inputFiles_R1="./trim_paired/input_files_R1.txt"
array_R1=($(<$inputFiles_R1))
mkdir -p bam_without_mitochondria
mkdir -p picard_insert_size_after_mito_dup
mkdir -p bam_without_mito_dup
mkdir -p reads_per_chromosome
mkdir -p picard_duplicates

for SAMPLE in {0..5}
```

```
do
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%_L1*}
echo ${FL1_trimmed}

echo "removing mitochondrial chromosome from ${FL1_trimmed}"


samtools idxstats -@ 10 ./SAM_files/${FL1_trimmed}.sorted.bam  |  cut -f 1 |  grep -v -e MT \
-e Citrine_N1 -e Clover_GFP -e dsRed -e Egfp -e Egfp_Kozak_3_UTR -e FynTyrosinase -e mCherry \
-e mEGFP -e mKate2 -e mRuby2 -e PHluorin -e TagRFP_T -e tdTomato -e tdTomato_silent_SacI -e Turquoise \
-e YFP | xargs samtools view --threads 10 -m 4G \
-b ./SAM_files/${FL1_trimmed}.sorted.bam > ./bam_without_mitochondria/${FL1_trimmed}_without_mito.bam

echo "creating ${FL1_trimmed}.bam index"
samtools index -@ 10 -m 4G ./bam_without_mitochondria/${FL1_trimmed}_without_mito.bam \
./bam_without_mitochondria/${FL1_trimmed}_without_mito.bam.bai

echo "counting reads per chromosome in ${FL1_trimmed}.without_mito.bam"

#write a file with reads per chromosome to check if unwanted chromosomes are removed or not
samtools idxstats bam_without_mitochondria/${FL1_trimmed}_without_mito.bam | \
awk '{print $1" "$3}' >> ./reads_per_chromosome/${FL1_trimmed}_reads_per_chromosome_without_mito.txt

#now running picard to remove duplicates
echo "removing duplicates in ${FL1_trimmed}"
java -jar /picard.jar MarkDuplicates I=./bam_without_mitochondria/${FL1_trimmed}_without_mito.bam \
O=./bam_without_mito_dup/${FL1_trimmed}_bam_without_mito_dup.bam \
M=./picard_duplicates/${FL1_trimmed}_dups.txt REMOVE_DUPLICATES=true

echo "making histograms for insert sizes for ${FL1_trimmed}"
# make histograms of insert sizes
java -jar /picard.jar CollectInsertSizeMetrics I=./bam_without_mito_dup/${FL1_trimmed}_bam_without_mito_dup.bam \
O=./picard_insert_size_after_mito_dup/${FL1_trimmed}_rm_mito_dup_insert_size_metrics.txt \
H=./picard_insert_size_after_mito_dup/${FL1_trimmed}_rm_mito_dup_insert_size_histogram.pdf
```

```
echo "counting reads per chromosome in ${FL1_trimmed}._bam_without_mito_dup.bam"
samtools idxstats bam_without_mito_dup/${FL1_trimmed}_bam_without_mito_dup.bam | \
awk '{print $1" "$3}' >> ./reads_per_chromosome/${FL1_trimmed}_reads_per_chromosome_without_mito_dup.txt
done
```

Get Fragment lengths again

```
#run picard to get fragment lengths

inputFiles_R1="./trim_paired/input_files_R1.txt"
array_R1=($(<$inputFiles_R1))
mkdir -p picard_insert_size_after_mito_dup

for SAMPLE in {0..5}
do
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%_L1*}
echo ${FL1_trimmed}

echo "starting picard on ${FL1_trimmed}"

java -jar /picard.jar CollectInsertSizeMetrics I=./bam_without_mito_dup/${FL1_trimmed}_bam_without_mito_dup.bam \
O=./picard_insert_size_after_mito_dup/${FL1_trimmed}_rm_mito_dup_insert_size_metrics.txt \
H=./picard_insert_size_after_mito_dup/${FL1_trimmed}_rm_mito_dup_insert_size_histogram.pdf

done
```

Downsampling

```
function SubSample {

## see also: http://crazyhottommy.blogspot.com/2016/05/downsampling-for-bam-files-to-certain.html
FACTOR=$(samtools idxstats $1 | cut -f3 | awk -v COUNT=$2 'BEGIN {total=0} {total += $1} END {print COUNT/total}')

if [[ $FACTOR > 1 ]]
  then
  echo '[ERROR]: Requested number of reads exceeds total read count in' $1 '-- exiting' && exit 1
fi

sambamba-0.8.0 view -s $FACTOR -t 2 -f bam -l 5 $1

}

export -f SubSample

ls ./bam_without_mito_dup/*.bam | parallel "SubSample {} 11082464 > {.}_subsampled.bam"
```

Take only unique reads using samtools

```
inputFiles_R1="./trim_paired/input_files_R1.txt"
array_R1=($(<$inputFiles_R1))
mkdir -p only_unique_downsampled
mkdir -p picard_insert_size_unique_only_downsampled

for SAMPLE in {0..5}
do
```

```
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%_L1*}
echo ${FL1_trimmed}

echo "removing non unique reads from ${FL1_trimmed}"

samtools view --threads 10 -m 4G -b -q 10 ./bam_without_mito_dup/${FL1_trimmed}_bam_without_mito_dup_subsampled.bam \
> ./only_unique_downsampled/${FL1_trimmed}_only_unique_downsampled.bam

echo "making histograms for insert sizes for ${FL1_trimmed}"
# make histograms of insert sizes
java -jar /picard.jar CollectInsertSizeMetrics -I ./only_unique_downsampled/${FL1_trimmed}_only_unique_downsampled.bam \
-O ./picard_insert_size_unique_only_downsampled/${FL1_trimmed}_only_unique_downsampled_insert_size_metrics.txt \
-H ./picard_insert_size_unique_only_downsampled/${FL1_trimmed}_only_unique_downsampled_histogram.pdf

done
```

Get the genome size

```
# to get the genome size I used the following command:
./faCount -summary ./Ensmbl/GRCz11/v102/Danio_rerio.GRCz11.dna_sm.primary_assembly_fp.fa \
> ./Ensmbl/GRCz11/v102/faCount_summary.txt
```

Make bigWig files with nucleosome distribution profile

```
inputFiles_R1="./trim_paired/input_files_R1.txt"
array_R1=($(<$inputFiles_R1))
mkdir -p bigwig_rpgc_all_incl_downsampled
mkdir -p logs

NOCOLOR='\033[0m'
RED='\033[0;31m'
GREEN='\033[0;32m'
ORANGE='\033[0;33m'
BLUE='\033[0;34m'
PURPLE='\033[0;35m'
CYAN='\033[0;36m'
LIGHTGRAY='\033[0;37m'
DARKGRAY='\033[1;30m'
LIGHTRED='\033[1;31m'
LIGHTGREEN='\033[1;32m'
YELLOW='\033[1;33m'
LIGHTBLUE='\033[1;34m'
LIGHTPURPLE='\033[1;35m'
LIGHTCYAN='\033[1;36m'
WHITE='\033[1;37m'

echo -e "${ORANGE}bigwig_rpgc_all_incl${NOCOLOR}"

for SAMPLE in {0..5}
do
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%_L1*}
echo -e "${LIGHTBLUE}${FL1_trimmed}${NOCOLOR}"

bamCoverage -b ./only_unique_downsampled/${FL1_trimmed}_only_unique_downsampled.bam \
-o ./bigwig_rpgc_all_incl_downsampled/${FL1_trimmed}_only_unique_downsampled.bw \
--binSize 1 \
--normalizeUsing RPGC \
--effectiveGenomeSize 1373484369 \
--ignoreDuplicates \
```

```
-p 8 2> ./logs/${FL1_trimmed}_only_unique_downsampled.bigwig_rpgc_all_incl.log
done
```

Peak calling using Generich

```
#run genrich on bam files

inputFiles_R1="./trim_paired/input_files_R1.txt"
array_R1=($(<$inputFiles_R1))
mkdir -p Generich_downsampled
mkdir -p sorted_qname_bam_downsampled

for SAMPLE in {0..5}
do
#set input file 1 to "FL1", input file 2 to "FL2"
FL1=${array_R1[SAMPLE]}
#echo ${FL1}
FL1_trimmed=${FL1%_L1*}
echo ${FL1_trimmed}



# create queryname sorted file
samtools sort -n --threads 10 -m 4G ./only_unique_downsampled/${FL1_trimmed}_only_unique_downsampled.bam \
-o ./sorted_qname_bam_downsampled/${FL1_trimmed}_qname_sorted_downsampled.bam


#call Genrich command
PATH=~/Genrich:$PATH

./Genrich/Genrich  -t ./sorted_qname_bam_downsampled/${FL1_trimmed}_qname_sorted_downsampled.bam \
-o ./Generich_downsampled/${FL1_trimmed}_downsampled.narrowPeak \
-f ./Generich_downsampled/${FL1_trimmed}_bedgraph_pval_downsampled.bdg \
-k ./Generich_downsampled/${FL1_trimmed}_bedgraph_pileup_pval_downsampled.bdg \
```

```
-b ./Generich_downsampled/${FL1_trimmed}_downsampled.bed -j -v -e 1373484369

done
```