Introduction to ChIP-seq data analysis

Shuo Zhang
Penn Epigenetics Institute
03/20/2024

Description

This tutorial is designed to help you get familiar with ChIP-seq data analysis. We will describe methods to perform quality control for the raw fastq files, read mapping, and peak calling. We will also learn to create tracks for Integrative Genomics Viewer (IGV). Finally, we will create profile plots at specific genomic regions.

To learn these processes, we will use a dataset that compares H3K27me3 signals between normal diet and high-sugar diet in fruit fly:

Yang, Jie, et al. "Exposure to high-sugar diet induces transgenerational changes in sweet sensitivity and feeding behavior via H3K27me3 reprogramming." *Elife* 12 (2023): e85365.

Prerequisites

- A personal computer (Windows, MacOS, or Linux)
- A HPC account

Download the dataset

After logging in HPC, you can ask for a private note and download the dataset from github

```
$ bsub -Is bash
$ git clone https://github.com/szhang32/ChIPseq_tutorial.git
```

Install required software using conda, a package and environment management tool.

Check the version of conda:

```
$ conda --version
```

Install macs2:

```
$ conda create --name macs2          # create a conda environment named macs2
$ conda activate macs2               # activate macs2 environment
$ conda install -c bioconda macs2    # install macs2
$ macs2 callpeak -h                  # test if macs2 is installed successfully
$ conda deactivate                   # deactivate macs2 environment
```

Install deeptools:

```
$ conda create --name deeptools
$ conda activate deeptools
$ conda install bioconda::deeptools
$ conda deactivate
```

Install homer:

```
$ conda create --name homer
$ conda activate homer
$ conda install bioconda::homer
$ conda deactivate
```

Step 1: quality control for fastq files

1a. raw fastq quality control using FastQC. To finish running the command in a reasonable time, only first 250K read pairs are used.

```
$ module load FastQC-0.11.2
$ mkdir raw_fastqc                    # make a directory to store fastqc results
$ cd raw_fastqc
$ fastqc -o . ../../raw_fastq/F1_K27me3_rep1_R1.fq.gz
$ fastqc -o . ../../raw_fastq/ F1_K27me3_rep2_R1.fq.gz
```

- -o: output directory
- '.': current working directory
- '..': parent directory of current working directory

1b. Preprocess fastq files. We will use fastp: https://github.com/OpenGene/fastp, which performs quality control, remove bad reads, trim adapters etc.

```
$ mkdir preprocessed_fastq
$ cd preprocessed_fastq
$ /project/epi-gen-traininglab/software/fastp -f 10 -F 10 \
        -i ../../raw_fastq/F1_K27me3_rep1_R1.fq.gz \
        -I ../../raw_fastq/F1_K27me3_rep2_R1.fq.gz \
        -o F1_out_R1.fq.gz \
        -O F1_out_R2.fq.gz
```

- '\': allows one command to span multiple lines
- -f: trim # bases from the beginning of read1
- -F: trim # bases from the beginning of read2
- -i: input read1
- -I: input read2
- -o: output read1

- -O: output read2

1c. Quality control using FastQC as in 1a.

```
$ cd preprocessed_fastq
$ fastqc *.gz
```

- '*': a wildcard. *.gz represent all files ending with .gz

Step 2: alignment with bowtie2

2a. First, we need to build an index for mapping. To save time, we will only build an index for chr2L of the dm6 reference genome.
The chr2L.fa is stored in the ref directory. We will put index in the ref directory as well.

```
$ module add bowtie2/2.3.4.3
$ bowtie2-build chr2L.fa chr2L
```

two required parameters: 1. Reference 2. dir/basename

2b. Then, we can map the reads to the reference.

```
$ mkdir bam
$ cd bam
bowtie2 -q --no-mixed --no-unal --phred33 -x ../ref/chr2L -1 ../ preprocessed_fastq
/F1_out_R1.fq.gz -2 ../ preprocessed_fastq /F1_out_R2.fq.gz -S
F1_H3K27me3_rep1.sam
```

-q: reads are in FASTQ format
--no-mixed: don't find alignments for individual mates
--no-unal: don't output reads that fail to align
--phred33: phred score + 33 encoding
-x: basename of the index
-1 and -2: reads
-S: write output to SAM file

More information about bowtie2 parameters: https://bowtie-bio.sourceforge.net/bowtie2/manual.shtml

Step 3: filtering

3a. We will use samtools to low-quality reads.

```
$ module add samtools/1.9
$ samtools view -bS -q 20 F1_H3K27me3_rep1.sam > F1_H3K27me3_rep1.bam
$ samtools sort F1_H3K27me3_rep1.bam > F1_H3K27me3_rep1.sorted.bam
$ samtools index F1_H3K27me3_rep1.sorted.bam
```

More information about samtools: http://www.htslib.org/doc/samtools.html

3b. remove duplicates using picard:

```
$ module add picard/2.23.3
$ java -jar "$PICARD/picard.jar" MarkDuplicates I="F1_H3K27me3_rep1.sorted.bam"
O="F1_H3K27me3_rep1.RMDUP.bam" M=dup.txt REMOVE_DUPLICATES=true
```

We can also remove blacklist regions using bedtools (https://bedtools.readthedocs.io).

Step 4: call peak with MACS2

One essential step of ChIP-seq data analysis is to identify genomic regions that are enriched with signals of interest, such as histone modifications and transcription factors. MACS (Model-based Analysis of ChIP-seq) is a commonly used to call peaks for ChIP-seq data.

We can run macs2:

```
$ conda activate macs2
macs2 callpeak -t F1_H3K27me3_rep1.RMDUP.bam -f BAMPE -g 23000000 --broad --
broad-cutoff 0.01 --outdir . -n F1_H3K27me3_rep1
```

-t: treatment alignment file, can be in different format.
-f: alignment file format, BAMPE is paired-end bam
-g: mappable genome size, there are recommended sizes for model organisms
--broad: H3K27me3 has broad peaks
--broad-cutoff: cutoff for the broad region
--outdir: output directory
-n: output name (prefix)

For more information about MACS: https://github.com/macs3-project/MACS
As too few reads (250K) are used, it is no surprise that no broad peak is identified.

After finish calling peaks, we can exit the macs2 environment by:

```
$ conda deactivate
```

Step 5: create an IGV track with deeptools

Visualizing ChIP-seq data is a great way to examine the data quality and find biological patterns. Here, we will create a track to be upload to IGV or genome browser. We can accomplish that using deeptools.

the bamCoverage function converts a .bam file to a .bigwig or .bedgraph file

```
$ bamCoverage -b F1_H3K27me3_rep1.RMDUP.bam -o F1_H3K27me3_rep1.bw --
normalizeUsing CPM
```

-b: bam file input
-o: output file name
--normalizeUsing: normalization method, Count Per Million (CPM)

Step 6: create profile plots with deeptools

For this task, we will plot H3K27me3 signal around regions that have higher H3K27me3 enrichment in high-sugar diet compared to control. Published data (in the .bw directory) include

GSM6658146_ND_H3K27me3_1.bw: H3K27me3 in control replicate 1
GSM6658159_F1_H3K27me3_1.bw: H3K27me3 in high-sugar diet
chr2L_rep1_up_peaks.bed: up-regulated H3K27me3 regions

We first compute the signal:

```
$ computeMatrix scale-regions -S GSM6658146_ND_H3K27me3_1.bw
GSM6658159_F1_H3K27me3_1.bw -R chr2L_rep1_up_peaks.bed -a 2500 -b 2500 -o
result.mat.gz --startLabel Start --endLabel End  --samplesLabel ND F1
```

-S: bigwig files
-R: regions
-a: distance after region end
-b: distance before region start

Then, we plot the signal:

```
$ plotProfile -m result.mat.gz -o result.pdf --perGroup --plotTitle "H3K27me3" --colors
gray pink --startLabel Start --endLabel End
```

-m: input matrix
-o: output file name
--perGroup: group samples for each region
--plotTitle: title for the plot
--colors: colors for each sample
--startLabel: start label
--endLabel: end label

For more information about deeptools:
https://deeptools.readthedocs.io/en/develop/index.html