Data analysis of a ChIP-seq experiment with two replicates and multiple immunoprecipitated proteins

Author: Tomàs Montserrat Ayuso

December 18, 2022

Introduction

In this new post on analyzing data from ChIP-seq experiments we will follow the Galaxy Formation of the Super-Structures on the Inactive X tutorial. However, we will perform all the steps from the terminal. We will also do a couple more tasks to illustrate how we can handle the results of different replications.

The data we will use in this post comes from a study carried out by Wang et al. 2018 and aimed to study the differences in the DNA binding sites of histones H3K27me3, H3K4me3 and the transcription factor CTCF between two different experimental conditions (wild-type and SMCHD1 knockdown). We will simply use the data from the wild-type condition to exemplify the analysis process of this type of data, from the time we obtain the reads in *fastq* format until we obtain the DNA binding regions of these annotated proteins. In our case, the annotation will be based on finding the genes closest to these regions.

The pipeline we will follow in this post is as follows:

- Quality control of the reads of *fastq* files with the *fastqc* program.
- Alignment of high-quality reads against the reference genome with the *bowtie2* program.
- Study of the quality of the ChIP-seq experiment studying the correlation between the samples and the effectiveness of the immunoprecipitation in each sample. We will do this step with the tools of the *deepTools* suite.
- Normalization of the samples to make them comparable with the control samples. We will do this with two different methods: normalizing by sequencing depth and depending on the control sample (input).
- Detection of the genome regions differentially enriched in reads (peak calling) with *macs2*.
- Comparison of the signal of each sample in the peaks found from a heatmap generated with tools from the *deepTools* suite.
- Find the shared peaks between the two replicates of each sample using *bedtools intersect*.
- Annotate the peaks by looking for the genes closest to them with *bedtools closest*.

The data files for this project can be downloaded by following this link.

Project directories

For this post, we will need a directory structure similar to this:

```
.
├── alignment
│   ├── bam_files
│   ├── input_normalized_coverage
│   ├── normalized_coverage
│   └── sequencing_depth
├── annotations
├── data
├── macs
├── quality
├── reference
├── results
│   ├── closest_genes
│   ├── concatenated_peaks
│   ├── heatmap
│   ├── merged_peaks
│   ├── region_scores_matrix
│   └── treatments_peaks
└── scripts
```

Given that during the analysis we will generate quite a few intermediate files, it is convenient to be tidied in our working directory. In the *data* folder we will save the raw data, that is, the *fastq* files. In *quality*, the different reports, graphs and files with information about the quality of the samples. In *alignment*, all the files related to the alignment. In *reference*, we will save the reference genome and its index. In *annotations*, we will save the genome annotation file that we will use. In *results*, we will save the different results we will get. Finally, in the *scripts* folder we will save the script with all the commands to replicate the analysis.

Quality control of the reads

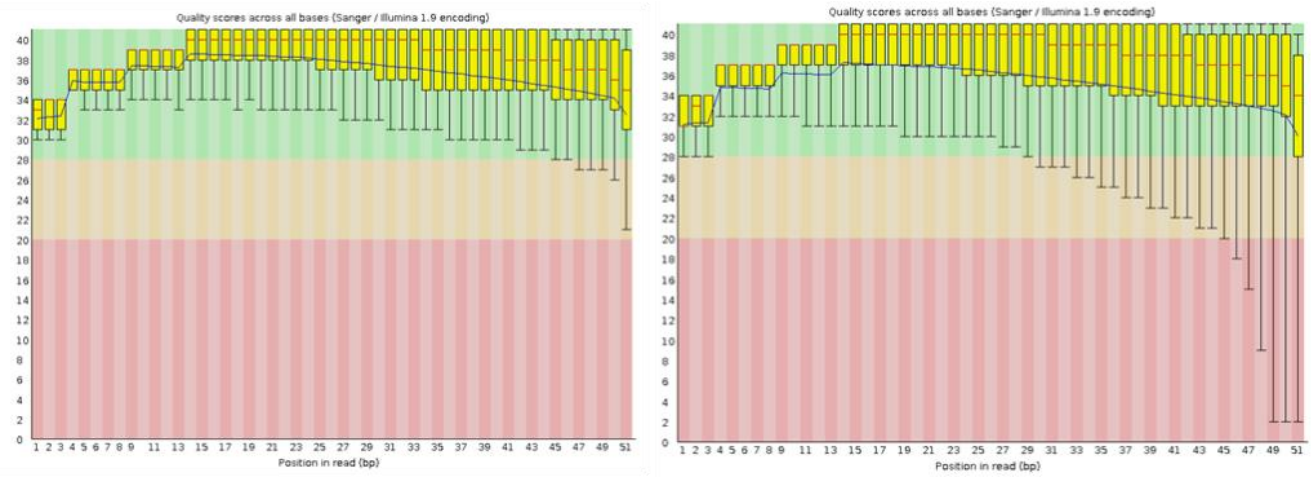
We only have the *fastq* files, with 50,000 reads each, for the H3K4me3 protein immunoprecipitation samples. Therefore, we will exemplify data pre-processing (quality control and alignment) with these files.

We have paired-end sequencing data, which means that we have sequenced both ends of each DNA fragment. Therefore, we need to perform quality control on the reads of the forward chain from one end of the fragment and the reads of the reverse chain from the other end of the fragment. To do this quality control we will use the

program called *fastqc* which takes as many *fastq* files as we want and returns a report for each of them detailing different aspects related to their quality.

```
$ fastqc data/wt_H3K4me3_read1.fastq data/wt_H3K4me3_read2.fastq -o quality/
```

We will focus on the quality distribution for each position of the reads. For more information on the quality reports generated by this program you can consult this article or this web page. On the left we have the quality of the sequences of the first read and on the right that of the second:



We can see that the quality of the samples is very high for all positions and that they follow the typical pattern in sequencing samples in that the confidence that the reported base is correct increases during the first positions and decreases for the last. Other indicators that can be consulted in the report also inform us of the high quality of the reads: there are practically no duplicate sequences or adapter sequences.

Filtering of positions with low quality

Despite the high quality of the reads, we will perform a trimming of the last positions of the reads as an example. To do this, we will use the program *trimmomatic*:

```
$ trimmomatic PE -threads 1 -phred33 \
> data/wt_H3K4me3_read1.fastq data/wt_H3K4me3_read2.fastq \
> data/wt_H3K4me3_read1_trimmed_paired.fastq \
> data/wt_H3K4me3_read1_trimmed_unpaired.fastq \
> data/wt_H3K4me3_read2_trimmed_paired.fastq \
> data/wt_H3K4me3_read2_trimmed_unpaired.fastq \
> TRAILING:20 MINLEN:20
```

When we use *trimmomatic* we must specify that we have paired-end reads using the `PE` option. As an output we must indicate a total of four files: two for the forward reads and two for the reverse complement reads (one that collects all the paired reads and another that groups the reads that have lost the pair due to trimming). With the

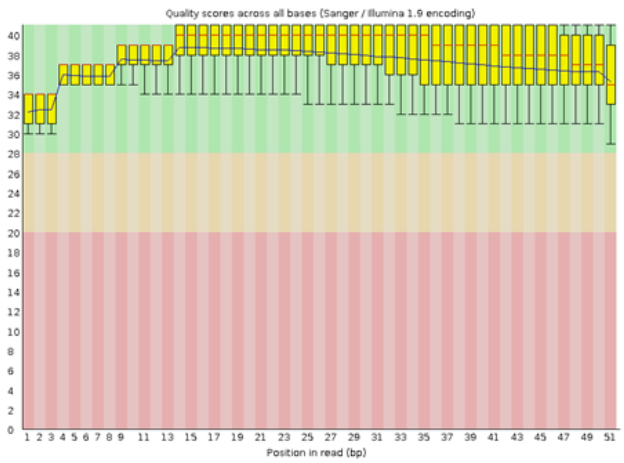
options `TRAILING:20` and `MINLEN:20` we ask that the sequences be trimmed by the "tail" while the quality of this position is below 20 and that the sequences with a length lower than 20 be removed.

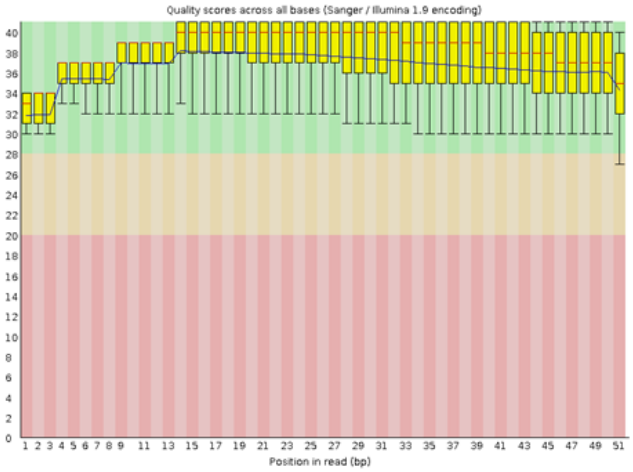
Quality control of post-trimming reads

Once the sequences have been cut, it is necessary to check that the final quality of the sequences is the desired one. Therefore, we generate the quality report again with *fastqc*, but this time on the trimmed *fastq* files:

```
$ fastqc data/wt_H3K4me3_read1_trimmed_paired.fastq \
> data/wt_H3K4me3_read2_trimmed_paired.fastq -o quality/
```

Inspection of the report confirms the removal of low-quality bases from the "tail" of the sequences:





Alignment of the reads with the reference genome

Satisfied with the quality of our reads, we can now perform the alignment on the reference genome to know which part of the genome they come from. We will use the program called *bowtie2*.

First, we need to download the *mm10* reference genome in *fasta* format. We can do it with the following command:

```
$ wget http://hgdownload.cse.ucsc.edu/goldenpath/mm10/bigZips/mm10.fa.gz \
> -O reference/mm10.fa.gz
$ gunzip reference/mm10.fa.gz
```

Before starting the alignment, however, we need to create the index for the reference genome using the program called *bowtie2-build*:

```
$ bowtie2-build reference/mm10.fa reference/mm10
```

Finally, we can align our reads:

```
$ bowtie2 -p 1 -q \
> -x reference/mm10 \
> -1 data/wt_H3K4me3_read1_trimmed.fastq -2 data/wt_H3K4me3_read2_trimmed.fastq \
> -S alignment/wt_H3K4me3.sam
```

The `-p` option indicates the number of processors we want to use, `-q` that the input are *fastq* files and `-x` the directory and index prefix of the reference genome. In the `-1` option we specify the forward read and in the `-2` option the reverse complement read. Finally, the `-S` option is used to specify that we want the output in *sam* format.

To finish this section, we will sort the sequences in our *sam* file, convert it to *bam*, and create an index of those files, which is necessary for many of the programs that will use *bam* files as input. We will do all this with programs from the *samtools* suite: *view* and *sort* to sort and convert to *bam*, and *index* to index the file:

```
$ samtools view -b alignment/wt_H3K4me3.sam | samtools sort -o
> alignment/wt_H3K4me3.bam
> samtools index -b alignment/wt_H3K4me3.bam
```

We will also index the rest of the *bam* files from the tutorial:

```
$ ls alignment/bam_files/*.bam | xargs -n 1 -P 1 samtools index -b
```

Quality control of the ChIP-seq experiment

At this point we will work with the *bam* files supplied by the Galaxy tutorial which only have reads corresponding to the X chromosome.

In order to evaluate the quality of the ChIP-seq experiment we will carry out two tasks. First, we will look at the correlation between *bam* files; that is, the similarity in the alignment results between the samples. Here we expect replicates for the same protein to be more similar to each other than to the rest of the samples. Then we will study the strength of the signal in each sample. Here we will see how different the distribution of reads is in the immunoprecipitated samples compared to the control samples.

Correlation between samples

To study the correlation between the samples we will follow two steps, both with tools from the *deepTools* suite. To start we will use *multiBamSummary* to calculate the number of unique reads aligned to each nucleotide (read coverage) for a large number of regions of each of the *bam* files entered:

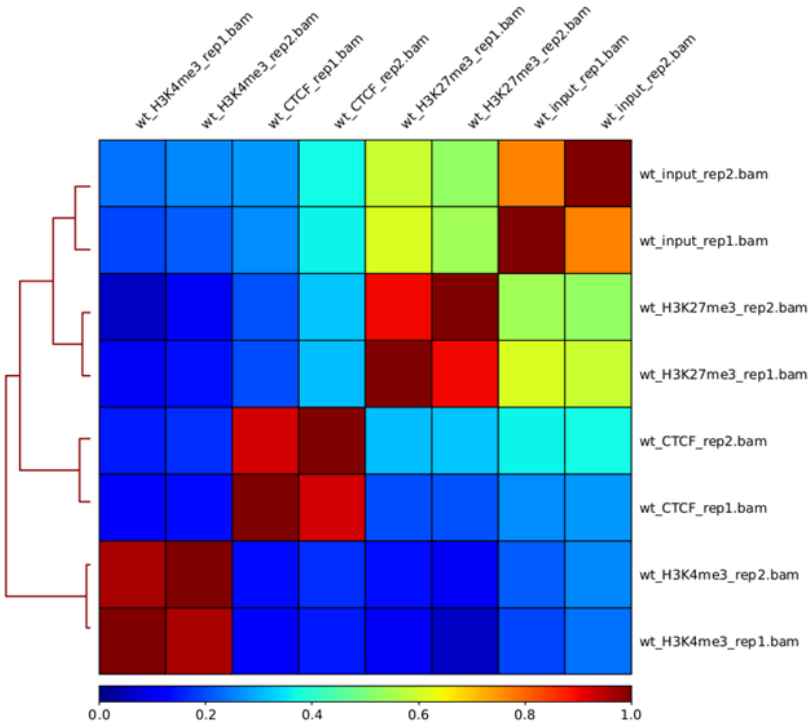
```
$  multiBamSummary bins --bamfiles alignment/bam_files/*.bam \
> --binSize 1000 \
> --distanceBetweenBins 500 \
> --region chrX \
> --outFileName quality/readCounts.npz \
> --outRawCounts quality/readCounts.tab
```

The `--binSize` option is the length of the nucleotide window used, while `--distanceBetweenBins` is the distance between one window and the next.

Once we have the *readCounts.npz* file we will use *plotCorrelation* to calculate and visualize the correlations between samples:

```
$ plotCorrelation --corData quality/readCounts.npz \
> --corMethod pearson \
> --whatToPlot heatmap \
> --plotFile quality/samples_correlation.pdf
```

The output of the program with the functions we have specified is a *pdf* file with the following heatmap:



As we could expect, the different replicates are more similar to each other than to the rest of the samples. The fact that the control samples are clustered together confirms that the immunoprecipitation has worked well in the experiment.

Estimation of the quality of the biological signal

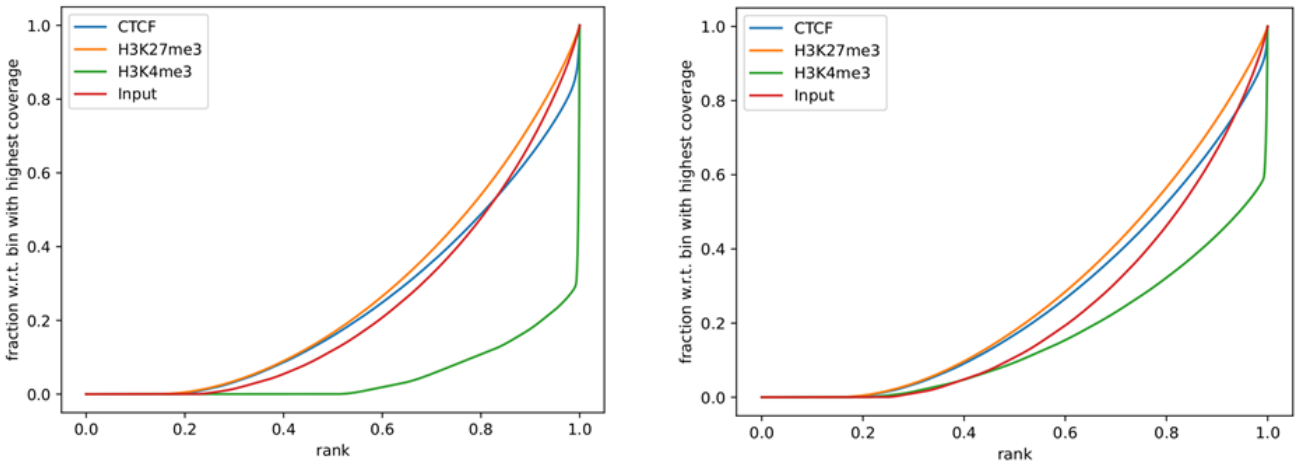
We will visualize the intensity of the biological signal in each sample by making a fingerprint plot with *plotFingerprint* of the *deepTools* suite. We will make one for each replicate:

```
$ INPUT1="alignment/bam_files/wt_input_rep1.bam"
$ INPUT2="alignment/bam_files/wt_input_rep2.bam"
$ SAMPLES1=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep1")
$ SAMPLES2=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep2")
```

```
$ plotFingerprint \
> -b ${SAMPLES1} ${INPUT1} \
> --labels CTCF H3K27me3 H3K4me3 Input \
> --region chrX \
> --numberOfSamples 10000 \
> --plotFile quality/fingerprint_rep1.pdf

$ plotFingerprint \
> -b ${SAMPLES2} ${INPUT2} \
> --labels CTCF H3K27me3 H3K4me3 Input \
> --region chrX \
> --numberOfSamples 10000 \
> --plotFile quality/fingerprint_rep2.pdf
```

The program will take 10,000 random regions of the genome (`--numberOfSamples`) and sum the per-base coverage of each *bam* file. These coverage values are sorted and the accumulated sum is displayed. On the left is the graph for replicates 1 and on the right for replicates 2:



The interpretation of these graphs is quite simple. If the reads are distributed uniformly in the genome, we will get a straight line (this is what we expect to get in the control samples). If there are regions enriched in reads, then the distribution will not be homogeneous: there will be parts of the genome that will contain more reads than others. In our case only the immunoprecipitated sample for the H3K4me3 protein shows a strong biological signal. The rest of the samples do not differ from the controls, the signal seems very weak, especially for replicates 1 and the H3K27me3 protein.

Indeed, these results agree with the heatmap, where we observe that the control samples are more similar with H3K27me3 and CTCF in replicate 2. In general, we see that the immunoprecipitation has gone worse in the second replicate.

Normalization

In order to visually compare the accumulation of aligned reads in each position of the genome (the peaks), we need to normalize the data. We will see two ways to normalize the data: depending on the sequencing depth and depending on the input (control) file.

Normalization based on sequencing depth

We can estimate the sequencing depth of each sample using *idxstat* from the *samtools* suite in order to verify that the coverage of each *bam* file is different:

```
$ SAMPLES=$(ls alignment/bam_files/*.bam)
$ for i in $SAMPLES
> do
>       samtools idxstats $i > alignment/sequencing_depth/`echo $i | cut -d "/" \
>       -f 3 | cut -d "." -f 1`.txt
> done
```

If we take a look at a couple of the generated files we will see the difference in coverage:

```
$ cat wt_H3K4me3_rep1.txt | grep "X"
X       171031299       12048210

$ cat wt_input_rep1.txt | grep "X"
X       171031299       18935950
```

The second column of the file is the length of the chromosome, while the third is the number of reads aligned to that sequence.

The normalization of the reads based on the coverage of the entire genome is done using the *RPGC* method. To do this, we will use *bamCoverage* from the *deepTools* suite:

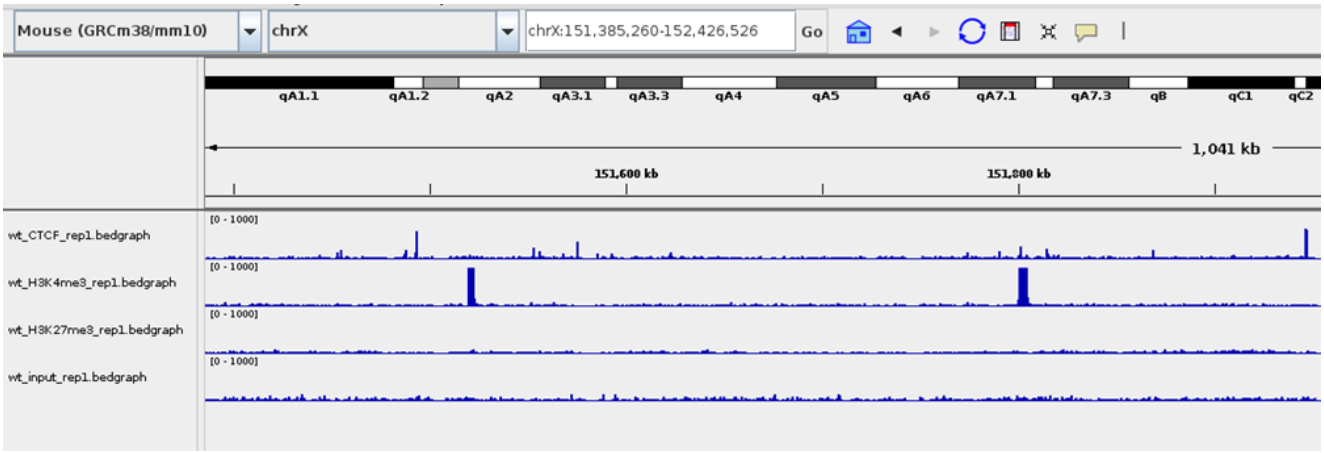
```
$ GENOMESIZE=2308125349
$ for i in $SAMPLES
> do
>       bamCoverage --bam $i \
>       --outFileName alignment/normalized_coverage/`echo $i | cut -d "/" -f 3 |\       >
cut -d "." -f 1`.bedgraph \
>       --outFileFormat bedgraph --normalizeUsing RPGC \
>       --effectiveGenomeSize $GENOMESIZE \
>       --binSize 25 \
>       --region chrX
> done
```

In the `--normalizeUsing` option we specified the *RPGC* method, which will use the genome size specified in the `--effectiveGenomeSize` option to perform the normalization. Effective genome size is defined as the

length of the genome that is alignable. In our case we used the effective size of the *GRCm38* version of the mouse genome provided on the *deepTools* website.

The `--binSize` option defines the number of nitrogenous bases that each segment of the generated *bedgraph* or *bigwig* file will have (in our case, we specified *bedgraph*).

If we view the *bedgraph* files in IGV, we will now see that all the samples seem to have similar coverage, as we can see in the following screenshot:



Normalization based on the input file

To compare the problem samples with the controls (input), normalization is usually performed based on the input file. To do this we will use *bamCompare* from the *deeptools* suite. This program takes both problem and control sample (replicate 1) bam files as inputs and returns a normalized *bedgraph* or *bigwig* file based on sequencing depth and with the log2 of the ratio of reads from each sample aligned at each position of the genome. Thus, positions where the problem sample has more aligned reads will have positive scores, while positions with fewer reads aligned to the problem sample will have negative scores.

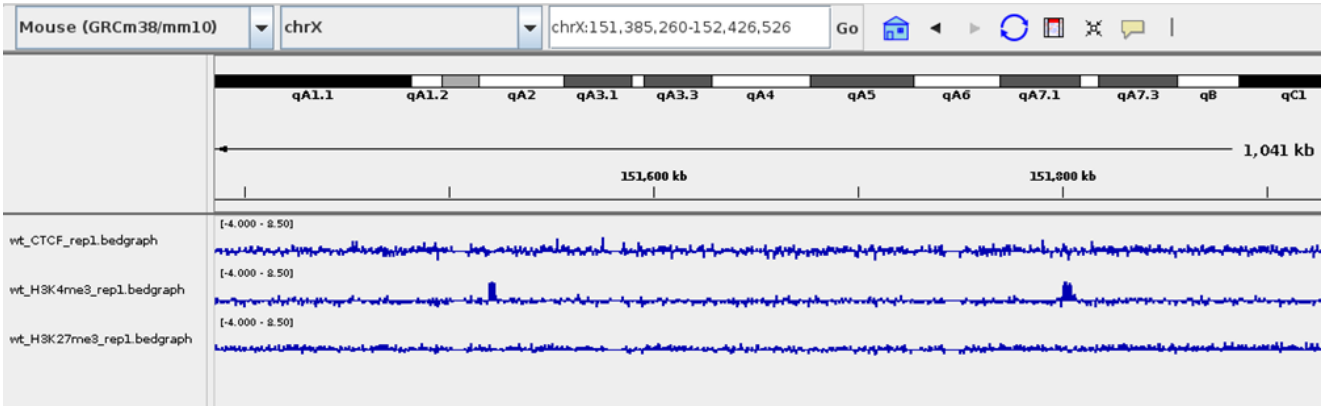
```
$ INPUT1="alignment/bam_files/wt_input_rep1.bam"
$ INPUT2="alignment/bam_files/wt_input_rep2.bam"
$ SAMPLES1=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep1")
$ SAMPLES2=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep2")
$ echo "bedgraph"
$ echo "Starting Rep 1"
$ for i in $SAMPLES1
>       do
>       bamCompare -b1 $i -b2 $INPUT1 \
>       --binSize 50 \
>       --operation log2 \
>       --outFileName alignment/input_normalized_coverage/`echo $i | cut -d "/" \
>       -f 3 | cut -d "." -f 1`.bedgraph \
>       --outFileFormat bedgraph \
>       --region chrX
```

```
$ done

$ for i in $SAMPLES2
$ do
>       echo " "
>       bamCompare -b1 $i -b2 $INPUT2 \
>       --binSize 50 \
>       --operation log2 \
>       --outFileName alignment/input_normalized_coverage/`echo $i | cut -d "/" \
>       -f 3 | cut -d "." -f 1`.bedgraph \
>       --outFileFormat bedgraph \
>       --region chrX
$ done
```

As with *bamCoverage*, the `--binSize` option specifies the number of nitrogenous bases that each group of reads in the resulting *bedgraph* file will have.

As an example, we will visualize the samples of the first replication in the same region of the X chromosome. With this type of normalization it is easy to see how in the sample the H3K4me3 protein has obtained better results in the immunoprecipitation:



Peak calling

On visual inspection we see some areas of the genome enriched in reads, especially in the H4K4me3 protein sample. Now we will perform peak calling, that is, we will use a specific program, *macs2*, to find all those regions of the genome that are significantly enriched in reads in the problem sample compared to the control. The following piece of code calls this program and returns a series of files with information about the peaks found:

```
$ SAMPLES1narrow=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep1" | grep -v
"H3K27me")
$ SAMPLES1broad=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep1" | grep
"H3K27me")
```

```
$ SAMPLES2narrow=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep2" | grep -v
"H3K27me")
$ SAMPLES2broad=$(ls alignment/bam_files/*.bam | grep -v "input" | grep "rep2" | grep
"H3K27me")

$ for i in $SAMPLES1narrow
> do
>       macs2 callpeak --treatment $i \
>       --control $INPUT1 \
>       --format BAMPE \
>       --gsize mm \
>       --name macs/macs_`echo $i | cut -d "/" -f 3 | cut -d "." -f 1`
> done

$ for i in $SAMPLES1broad
$ do
>       macs2 callpeak --treatment $i \
>       --control $INPUT1 \
>       --format BAMPE \
>       --gsize mm \
>       --broad \
>       --name macs/macs_`echo $i | cut -d "/" -f 3 | cut -d "." -f 1`
$ done
```

The same block of code, but replacing `$SAMPLES1` with `$SAMPLES2` and `$INPUT1` with `$INPUT2` will help us find the peaks of the samples of the second replicate.

By default, *macs2* works using a specialized algorithm to identify narrow peaks. According to the literature, H3K27me3 histones return broad peaks, so we specify the --broad option when we call *macs2* on the samples for this protein. Of the other options, the only thing to note is that, since we are working with paired reads, we must specify it using the `--format BAMPE` option. In this way, the program will not estimate the length of the fragments from which each read comes but will calculate it from the pairs of reads.

The two most important files returned by the program are **_peaks.narrowPeaks* and **_peaks.xls*. The first is a file in bed 6+4 form with the coordinates of the peaks found. The second is a tabular file with more information for each peak: the start, end and peak coordinates, the length, the height (pileup), the p-value, the q-value and the fold enrichment. Thanks to the *xls* file we can answer questions such as how many peaks were identified in the region `chrX:151,385,260-152,426,526` in the H3K4me3 protein sample in the first replicate:

```
$ awk '{ if(($2 >= 151385260) && ($3 <= 152426526)) {print $0} }' \
> macs_wt_H3K4me3_rep1_peaks.xls | wc -l
12
```

Now that we have the normalized alignment data in our *bedgraph* or *bigwig* files (to get the *bigwig* files we just have to specify it in *bamCoverage* and *bamCompare*) and the peaks found, we can do different downstream analyses. As an example, we will see how to compare the peaks found between two samples visually using a heatmap and how to annotate the peaks preserved between the different replicates by finding the closest genes.

Comparison of the signal between the samples

In order to compare the signal between the samples using a heatmap, we need to perform a series of previous steps:

- Concatenate, sort, and merge (merge) the coordinates of peaks found that overlap in two or more samples. We will do this with the programs *cat, sortBED* and *mergeBED.*
- Using the program called *computeMatrix* from the *deepTools* suite, calculate the signal of these regions in each sample using the *bigwig* files and save the results in a matrix.
- Visualize the matrix using a heatmap with *plotHeatmap* from the *deepTools* suite. We will compare the signal between the protein samples that originate narrow peaks, that is, the CTCF and H3K4me3 protein samples.

We will compare the signal between the protein samples that originate narrow peaks, that is, the CTCF and H3K4me3 protein samples. We will start by concatenating, ordering and joining the peaks found for these proteins in the two replicates:

```
$ NARROWPEAKS=$(ls macs/*.narrowPeak)
> cat ${NARROWPEAKS} \
> | cut -f 1-5 > results/concatenated_peaks/concatenated_narrowpeaks.bed

$ sortBed -i results/concatenated_peaks/concatenated_narrowpeaks.bed \
> > results/concatenated_peaks/concatenated_narrowpeaks_sorted.bed

$ mergeBed -i results/concatenated_peaks/concatenated_narrowpeaks_sorted.bed \
> > results/merged_peaks/merged_narrowpeaks.bed
```

With the peaks concatenated, ordered and joined, we can now calculate the signal intensity in each of these regions with the program *computeMatrix* using the *bigwig* files for the coverage of each sample in each region of the genome:

```
$ SAMPLES=$(ls alignment/input_normalized_coverage/*.bigwig | grep -v "H3K27me3")

$ computeMatrix reference-point \
> --regionsFileName results/merged_peaks/merged_narrowpeaks.bed \
> --scoreFileName ${SAMPLES} \
> --outFileName results/region_scores_matrix/matrix_narrowpeaks.gz \
```

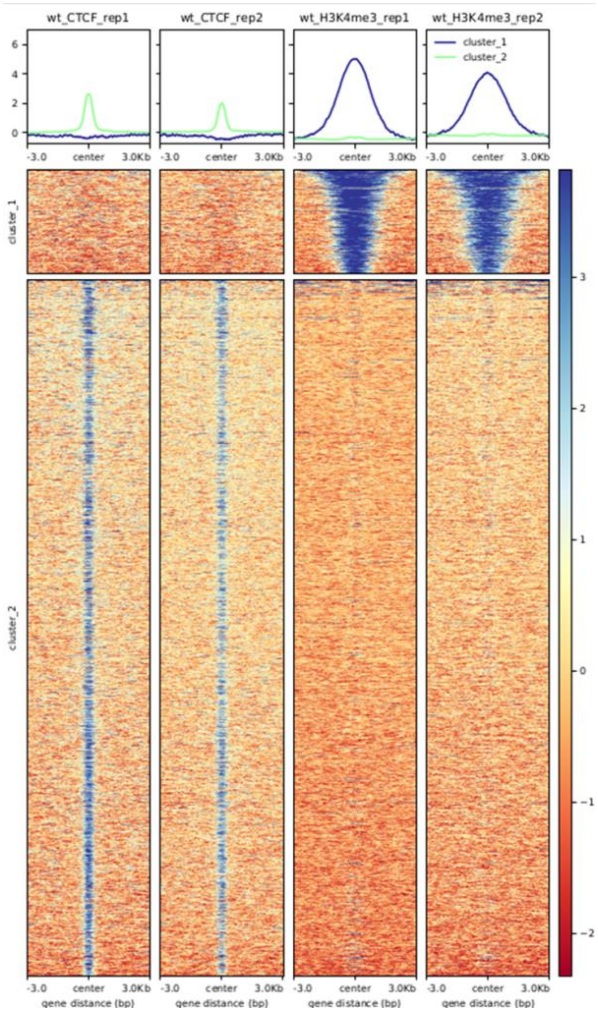
```
> --referencePoint center \
> --upstream 3000 \
> --downstream 3000
```

The `reference-point` option tells the program to use only the upstream and downstream positions of the regions indicated in the supplied bed file to calculate the scores. The number of upstream and downstream bases to consider are specified in the `--uptream` and `--downstream` options, respectively. The `--referencePoint` option allows us to specify what the reference point is for each region of the bed file; we selected `center` to indicate that we want the reference to be the center of each of these regions.

Finally, we can now create the heatmap with *plotHeatmap* to compare the two samples:

```
$ plotHeatmap -m results/region_scores_matrix/matrix_narrowpeaks.gz \
> --outFileName results/heatmap/heatmap_narrowpeaks.pdf \
> --refPointLabel center \
> --legendLocation upper-left \
> --kmeans 2 \
> --yMax 7
```

Based on the matrix data calculated in the previous step, this program returns the following heatmap:



Looking at the graph, we can conclude that fewer but larger peaks were found in the H3K4me3 sample. Having specified the `--kmeans 2` option, the program has divided the matrix into two clusters using the *k-means* algorithm. Thanks to this, we can see how the regions where peaks have been found for the transcription factor CTCF are different from the binding sites of the H3K4me3 protein.

Find the conserved peaks between the replicates and the genes closest to these peaks

To find the conserved peaks between the two replicas for each protein, we will use *intersect* from the *bedtools* suite which returns a *bed* file with the coordinates where the two files overlap:

```
$ bedtools intersect \
> -a macs/macs_wt_H3K4me3_rep1_peaks.narrowPeak \
> -b macs/macs_wt_H3K4me3_rep2_peaks.narrowPeak \
> > results/treatments_peaks/H3K4me3_overlaps.bed

$ bedtools intersect \
> -a macs/macs_wt_H3K27me3_rep1_peaks.broadPeak \
> -b macs/macs_wt_H3K27me3_rep2_peaks.broadPeak \
> > results/treatments_peaks/H3K27me3_overlaps.bed

$ bedtools intersect \
> -a macs/macs_wt_CTCF_rep1_peaks.narrowPeak \
> -b macs/macs_wt_CTCF_rep2_peaks.narrowPeak \
> > results/treatments_peaks/CTCF_overlaps.bed
```

Once we have the *bed* file with the peaks found in the two replicates for each protein, we will now look for the genes closest to each of these peaks. To achieve this, we need a genome annotation file (a *gtf* file) that contains the coordinates of all genes. Since we are only working on the X chromosome, we will filter this file to keep only the annotations of this chromosome and convert it to *bed* because *gtf* files are 1-based and bed 0-based. Using *closest* from the *bedtools* suite, we will find the genes closest to the coordinates of the conserved peaks.

Let's start by downloading the file with the annotations of each gene. We can do this using the following command:

```
$ wget https://hgdownload.soe.ucsc.edu/goldenPath/mm10/bigZips/genes/mm10.refGene.gtf.gz \
> -O annotations/mm10.refGene.gtf.gz
$ gunzip annotations/mm10.refGene.gtf
```

Now, we filter the genes to keep only those present on the X chromosome with an *awk* command:

```
$ awk '{ if ($1 == "chrX") { print } }' annotations/mm10.refGene.gtf \
> > annotations/mm10.refGene.chrX.gtf
```

Once we have the annotation file ready, we convert it to *bed* format with *gtf2bed* from the *bedops* suite:

```
$ gtf2bed < annotations/mm10.refGene.chrX.gtf > annotations/mm10.refGene.chrX.bed
```

Before using *closest* to find the genes closest to the peaks, we need to sort the peaks based on their coordinates and rename the chromosome in the annotation file from `chrX` to `X` to match the peaks file. First, we sort the peaks:

```
$ sort -k 1,1 -k2,2n results/treatments_peaks/CTCF_overlaps.bed \
> > results/treatments_peaks/CTCF_overlaps_sorted.bed

$ sort -k 1,1 -k2,2n results/treatments_peaks/H3K27me3_overlaps.bed \
> > results/treatments_peaks/H3K27me3_overlaps_sorted.bed

$ sort -k 1,1 -k2,2n results/treatments_peaks/H3K4me3_overlaps.bed \
> > results/treatments_peaks/H3K4me3_overlaps_sorted.bed
```

And now we rename the chromosome in the annotation file:

```
$ awk 'BEGIN { OFS = "\t" } $1="X"' annotations/mm10.refGene.chrX.bed \
> | cut -f 1-9 \
> > annotations/mm10.refGene.X.bed
```

Finally, we can now use *closest* to find the genes closest to the peaks:

```
$ bedtools closest -io \
> -a results/treatments_peaks/H3K4me3_overlaps_sorted.bed \
> -b annotations/mm10.refGene.X.bed \
> | cut -f 14 | sort | uniq \
> > results/closest_genes/closest_genes_H3K4me3.bed

$ bedtools closest -io \
> -a results/treatments_peaks/H3K27me3_overlaps_sorted.bed \
> -b annotations/mm10.refGene.X.bed \
> | cut -f 13 | sort | uniq \
> > results/closest_genes/closest_genes_H3K27me3.bed

$ bedtools closest -io \
> -a results/treatments_peaks/CTCF_overlaps_sorted.bed \
> -b annotations/mm10.refGene.X.bed \
> | cut -f 14 | sort | uniq \
> > results/closest_genes/closest_genes_CTCF.bed
```

By default, the program returns the closest or overlapping genes in the `-b` file. We specified the `-io` option so that it ignores overlapping genes and returns only the closest but non-overlapping ones. Let's take a look at the results for the H3K4me3 protein sample:

```
$ tail closest_genes_H3K4me3.bed
Zfp280c
Zfp300
Zfp449
Zfp711
Zfx
Zic3
Zmym3
Zrsr2
Zxda
Zxdb
```

Conclusions

In this entry we have seen how to carry out the data analysis of a ChIP-seq experiment with two replicates and for different DNA-binding proteins. We performed the quality control of the reads and their alignment with the reference genome. We have seen two different ways of normalizing the number of reads aligned in each region of the genome to make all files comparable. We found the regions of the genome significantly enriched in reads for each of the immunoprecipitations with *macs2* and carried out two further analyses: visual comparison of the enriched areas between two different immunoprecipitations, and annotation of the peaks by finding the genes closest to them.

The final script with all the commands to reproduce the analysis is on the blog's GitHub.