Finding the DNA binding motifs of a transcription factor from a ChIP-seq experiment

Author: Tomàs Montserrat Ayuso

November 1, 2022

Introduction

ChIP-seq experiments make it possible to study the DNA sequences that bind to a certain transcription factor. In this post we will study how we can carry out the processing of data from such an experiment, from *fastq* files to the analysis of enriched motifs and cross-referencing these with a database of binding sites of transcription factors.

We will use data from a real ChIP-seq experiment: GSE41195. Specifically, we will focus on the sequencing data contained in SRX189773 and SRX189778. The experiment seeks to find the DNA binding sites of the *Escherichia coli* FNR transcription factor. Part of the information to write this post has been taken from this tutorial.

ChIP-seq experiments typically generate two *fastq* files: one containing reads with sequences enriched for the transcription factor binding site of interest, while the other containing reads with all DNA sequences attached to some protein (control or input). The comparison of the distribution of the aligned reads of the two files is what will allow us to find the most likely binding sites of the FNR protein.

We will study a basic pipeline for the analysis of this data:

- Quality control of the reads of our *fastq* files. We will do this using the *fastqc* program.
- Alignment of *fastq* files (with reads that have passed quality control) against the reference genome. We will perform this step with *bowtie2*.
- Quality control of the ChIP-seq experiment with the *plotFingerprint* program from the *deepTools* suite.
- Peak calling to find the most probable binding sites of the transcription factor under study using *macs3*.
- Visualization of the results in IGV.
- Discovery and study of the motifs present in the binding sequences of the transcription factor.

Let's get started.

Creation of the directories

Before starting our analysis, it's a good idea to have our working directory organized in order to properly categorize the files that will be generated during the pipeline. We will create 7 different directories:

```
$ mkdir adapters alignment data macs quality reference scripts
```

In each of these folders we will save the files related to a different source of information. In *adapters* we will save the sequences of the adapters that we have to remove from the reads. In the *alignment* folder we will save all the files with the alignment of the sequences of the *fastq* files, which we will save in the *data* folder. On *macs* we will save the peak calling results. In *quality* we will store all the results from the different quality controls that we will perform. In *reference* we will save everything related to the reference genome. Finally, in *scripts* is where we will save the files with the different commands and programs that we will write.

Quality control of the reads

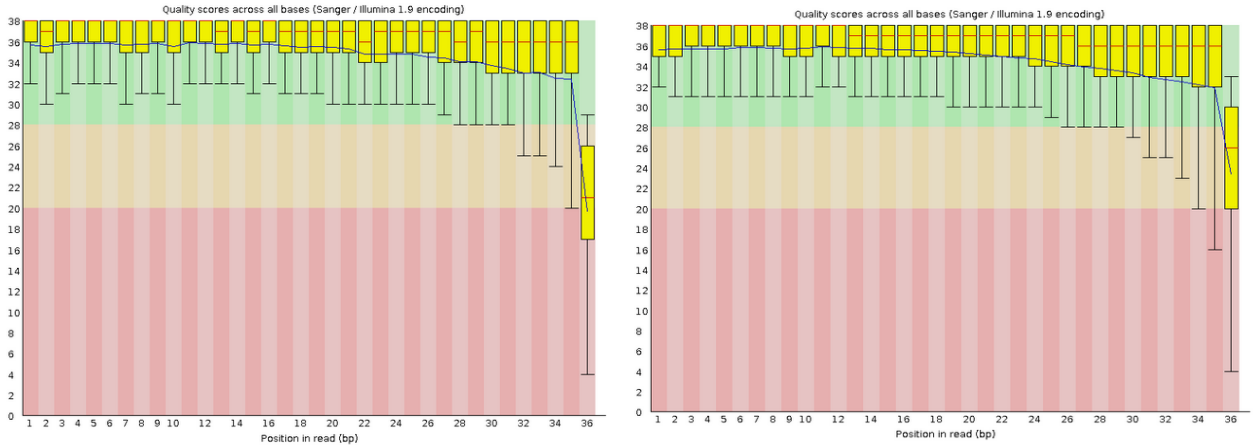
As with all NGS data analysis, the first thing we need to do is study the quality of the sequencing. The *fastqc* program will calculate several quality metrics of the reads and return a report so that we can evaluate whether the quality of the sequences is adequate or whether we need to do some pre-processing to eliminate low-quality reads and bases:

```
$ fastqc data/SRR576933.fastq -o quality/
```

```
$ fastqc data/SRR576938.fastq -o quality/
```

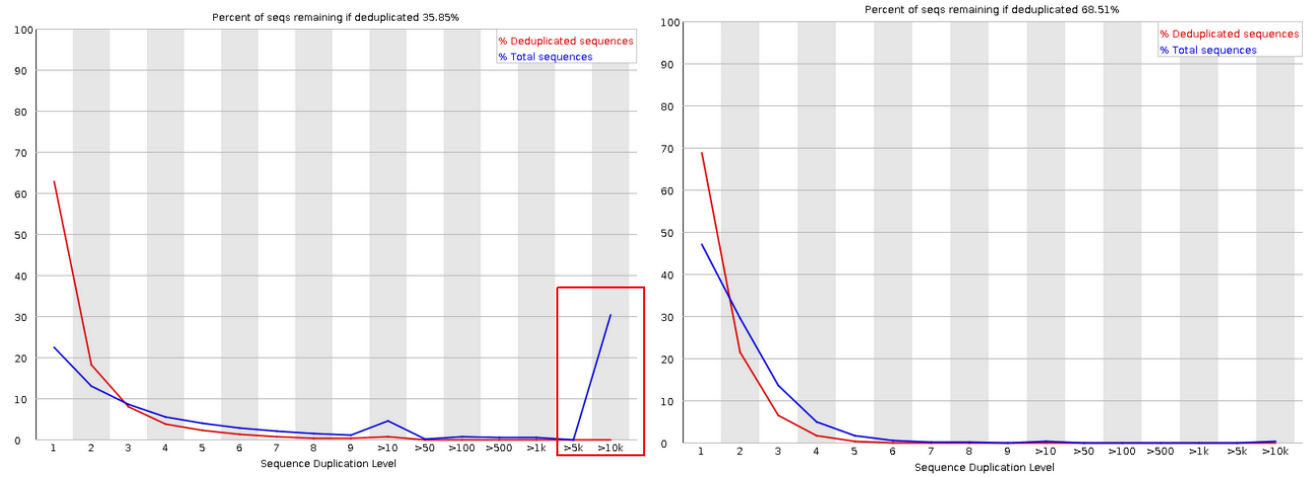
The `-o` option allows us to specify the directory where we want the report to be created. We repeated this process for the two samples: the control (*SRR576938.fastq*) and the problem (*SRR576933.fastq*). The generated report contains several sections. In this entry we will look at the quality distribution for each position of the reads and the presence of adapters. For more information on this report you can consult this previous article or this web page.

In the following figure we can see the quality distribution for each position. On the left, for the problem sample, and on the right for the control sample:



In general, the confidence that the reported base is correct is very high in all positions, except for the last one, where it decreases very much in general. This is a normal occurrence in sequencing data.

In the next plots we can inspect for the presence of adapters in the reads:



We see that in the problem sample we have an issue of duplicate reads. If we look at the frequency table of overrepresented sequences, we see that there is one that matches an adapter and that is found in a high percentage of the reads:

Sequence	Count	Percentage	Possible Source
GATCGGAAGAGCACACGTCTGAACTCCAGTCACACA	1060621	29.432719567181643	TruSeq Adapter, Index 5 (100% over 36bp)
GCTAACAAATACCCGACTAAATCAGTCAAGTAAATA	13630	0.37823875606902535	No Hit
NATCGGAAGAGCACACGTCTGAACTCCAGTCACACA	11728	0.3254573830651159	TruSeq Adapter, Index 5 (97% over 36bp)
GTTAGCTATTTACTTGACTGATTTAGTCGGGTATTT	10983	0.304783291115635	No Hit
GATCGGAAGAGCACACGTCTGAACTCCAGTCACACC	3658	0.10151117899490057	TruSeq Adapter, Index 1 (97% over 36bp)

Filtering of positions with low quality and removal of adapters

Once we have determined that the problem sample has an issue with the adapters, and that both samples have a quality problem at the last base of the sequences, we will use the *trimmomatic* program to trim the sequences by the tail if the reads do not reach a minimum quality score, and we will remove sequences that match (or are very similar to) those of the adapters. We will do this for both *fastq* files:

```
$ java -jar ${HOME}/trimmomatic/Trimmomatic-0.39/trimmomatic-0.39.jar SE \
> -threads 4 \
> data/SRR576933.fastq data/SRR576933_trimmed.fastq \
> ILLUMINACLIP:adapters/TruSeq3-SE.fa:2:30:10 \
> TRAILING:20 \
> MINLEN:20 \
> -phred33
```

```
$ java -jar ${HOME}/trimmomatic/Trimmomatic-0.39/trimmomatic-0.39.jar SE \
> -threads 4 \
> data/SRR576938.fastq data/SRR576938_trimmed.fastq \
> ILLUMINACLIP:adapters/TruSeq3-SE.fa:2:30:10 \
> TRAILING:20 \
> MINLEN:20 \
> -phred33
```

The `ILLUMINACLIP` option allows us to specify a file with the sequences of the adapters we want to remove. The numbers `2:30:10` establish the number of mismatches allowed to determine a match between the sequences of the adapter and those of the reads; the accuracy of the match between the two adapters in paired-end samples (does not apply to our case); and the minimum score of complete alignment between the adapter and the read for the adapter to be removed from the sequence.

With the `TRAILING` and `MINLEN` options we set the minimum quality that a base at the end of the read must have in order to keep it and the minimum number of bases that a sequence must have in order not to delete it, respectively.

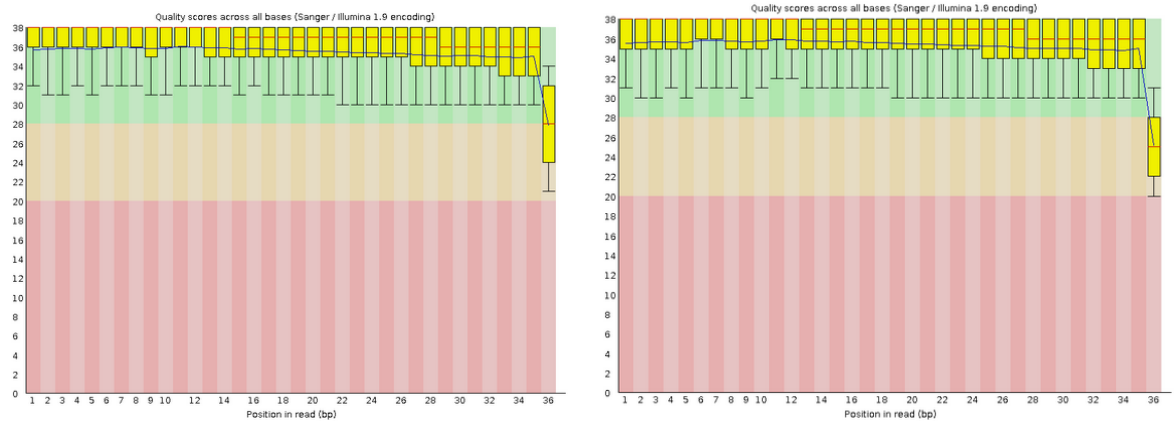
Quality control of trimmed reads

Once this step has been carried out, it is advisable to use the *fastqc* program again, this time with the *fastq* files created by *trimmomatic* to check that the quality of the reads is as desired:

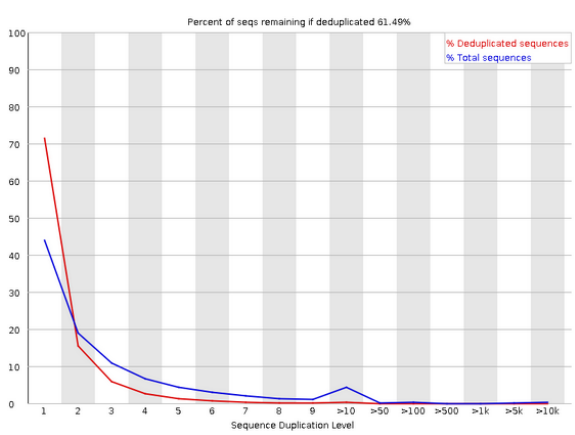
```
$ fastqc data/SRR576933_trimmed.fastq -o quality/
```

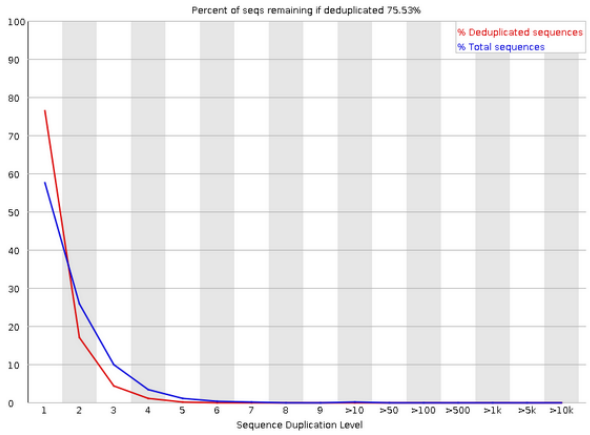
```
$ fastqc data/SRR576938_trimmed.fastq -o quality/
```

In the reports we can check how the last bases of bad quality reads have been removed:



And we no longer have the problem of adapters:





Finally, we will verify that we have enough coverage for our analysis. In general, about 10 million reads are usually needed for a ChIP-seq experiment with human samples (genome of about 3 Gb) to evaluate the access sites of transcription factors (narrow peaks). The *Escherichia coli* genome is much smaller, about 4.6 Mb. The first section of the report generated by *fastqc* tells us how many reads we have available:

Measure	Value
Filename	SRR576933_trimmed.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	2433914
Sequences flagged as poor quality	0
Sequence length	20-36
%GC	48

Measure	Value
Filename	SRR576938_trimmed.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	6606648
Sequences flagged as poor quality	0
Sequence length	20-36
%GC	49

The two samples (problem and control) give us sufficient coverage.

Satisfied with the quality, we can continue with our analysis.

Alignment of the sequences against the reference genome

The next step is to perform the alignment against the reference genome in *fasta* format. This file can be downloaded from this page. We will do this process using *bowtie2*. Before performing the alignment, however, we must build the genome index with the following command:

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

Once this is done, we can now perform the alignment for each of the *fastq* files:

```
$ bowtie2 -p 5 -q \
> -x reference/e_coli_genome \
> -U data/SRR576933_trimmed.fastq \
> -S alignment/SRR576933_unsorted.sam

$ bowtie2 -p 5 -q \
```

```
> -x reference/e_coli_genome \
> -U data/SRR576938_trimmed.fastq \
> -S alignment/SRR576938_unsorted.sam
```

The resulting files from the alignment are *sam* files which contain alignment information of each read. Next, we will transform the *sam* file into *bam* (which contains the same information but in binary form). We will also order the sequences by the coordinates of the genome where they have been aligned:

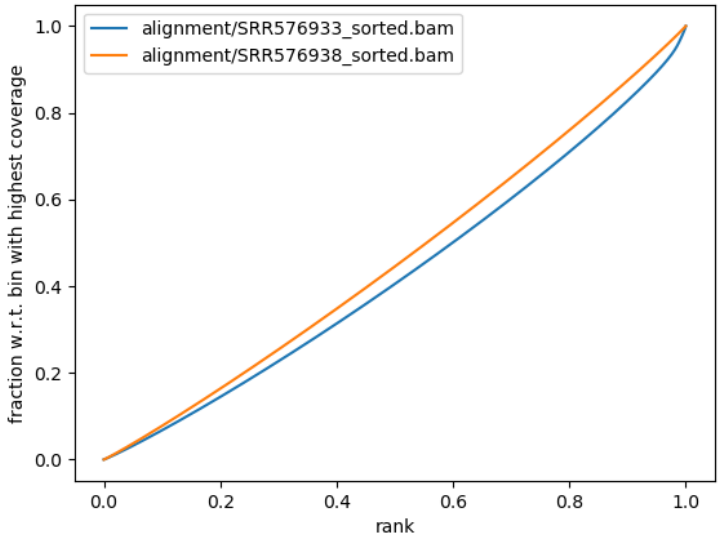
```
$ samtools view -b alignment/SRR576933_unsorted.sam | samtools sort -o
alignment/SRR576933_sorted.bam

$ samtools view -b alignment/SRR576938_unsorted.sam | samtools sort -o
alignment/SRR576938_sorted.bam
```

Quality control of the experiment

At this point, we will perform a basic quality control of the experiment by generating a *fingerprint* plot with the *plotFingerprint* program from the *deepTools* suite:

```
$ plotFingerprint -p 5 -b alignment/SRR576933_sorted.bam alignment/SRR576938_sorted.bam \
> -plot quality/fingerprint.png --ignoreDuplicates
```

The resulting plot is as follows:



The interpretation of this graph is quite simple. If the reads are distributed uniformly in the genome (what we expect from the control sample) we will get a straight line. If, on the other hand, there are regions enriched in reads (what we hope to find in a ChIP-seq experiment), then the distribution will not be homogeneous: there will be parts of the genome that will contain more reads than others. In our case the two lines are very similar, but not the same. It should be noted that since this is a ChIP-seq experiment to find binding motifs to a transcription

factor, it is normal that there are few and small enriched regions and weak signal, we will have what is known as narrow peaks.

Peak calling and visualization of the results

Once we have determined that the ChIP-seq experiment seems to have gone well and that there is a biological signal in the problem sample, we can proceed with peak calling. We will use the *macs3* program to perform this step. The software will search for regions of the genome enriched in reads, comparing the problem sample to the control. These regions will contain sequences that are very likely to be binding sites for the transcription factor we are studying. We can call this program with the following command:

```
$ macs3 callpeak -t alignment/SRR576933_sorted.bam \
> -c alignment/SRR576938_sorted.bam \
> --keep-dup 1 \
> --g $gsize \
> -n  macs/macs \
> -f BAM \
> -B \
> -q 0.05 \
> --nomodel \
> --extsize 211
```

The options we have specified are the following:

- `--keep-dup 1`: typically, only one of the duplicated reads is used.
- `-g $gsize`: we specify the effective size of the reference genome, stored in the variable `$gsize` (4,641,652, taken from https://www.ncbi.nlm.nih.gov/nuccore/556503834).
- `-n macs/macs`: prefix we will use in the name of the output files.
- `-f BAM`: format in which our reads are found.
- `-B`: *pileup* and *lambda* file format in *bedGraph*.
- `-q 0.05`: the minimum q-value to determine that a region is significantly enriched.
- `--nomodel`: option to specify that we don't want *macs* to estimate the size of sequenced chunks.
- `--extsize 211`: size of sequenced fragments. We determined it with the functions of the `ChIPQC` package from R.

The output of the program are different files, each with different information. The most important are the following:

- *macs_peaks.narrowPeaks*: is a 6+4 bed file containing the coordinates of the peaks.
- *macs_peaks.xls*: this tabular file contains information about the peaks. The initial, final and summit coordinates, length, height (pileup), p-value, q-value and fold enrichment.

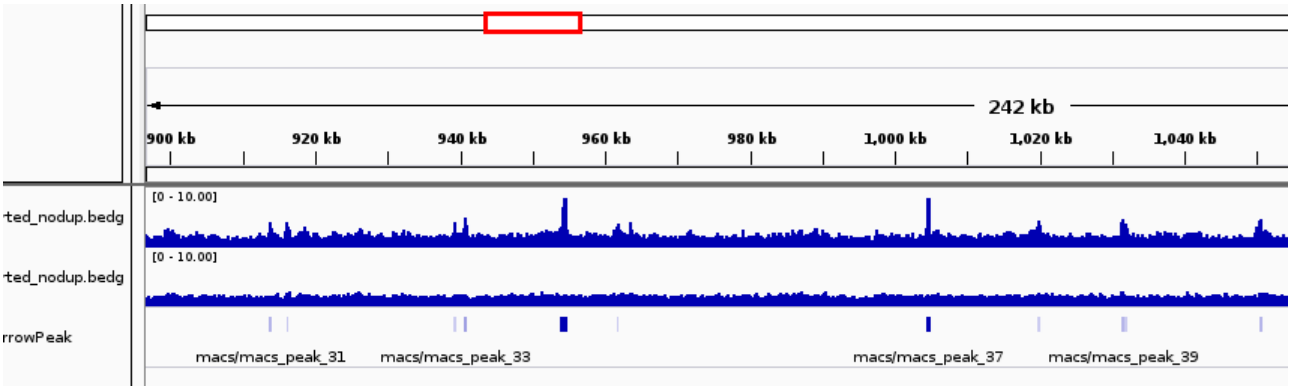
We will then view the results in *IGV*. Since the control and problem data have different coverage, we will generate a normalized *bedgraph* file from each *bam* file. In this way, neither of the two tracks (control and problem) will look bigger or smaller than the other. First, we'll remove duplicate reads and create an index of each *bam* file:

```
$ samtools rmdup -s alignment/SRR576933_sorted.bam \
> alignment/SRR576933_sorted_nodup.bam

$ samtools index alignment/SRR576933_sorted_nodup.bam \
> alignment/SRR576933_sorted_nodup.bai

$ samtools rmdup -s alignment/SRR576938_sorted.bam \
> alignment/SRR576938_sorted_nodup.bam

$ samtools index alignment/SRR576938_sorted_nodup.bam \
> alignment/SRR576938_sorted_nodup.bai
```

Finally, we normalize the data using the program called *bamCoverage* from the *deepTools* suite:

```
$ bamCoverage --bam alignment/SRR576933_sorted_nodup.bam \
> --outFileName alignment/SRR576933_sorted_nodup.bedgraph \
> --outFileFormat bedgraph --normalizeUsing RPGC \
> --effectiveGenomeSize $gsize

$ bamCoverage --bam alignment/SRR576938_sorted_nodup.bam \
> --outFileName alignment/SRR576938_sorted_nodup.bedgraph \
> --outFileFormat bedgraph --normalizeUsing RPGC \
> --effectiveGenomeSize $gsize
```

In the `--normalizeUsing` option we have specified the *RPGC* method, which will normalize the number of reads based on the coverage of the entire genome. We will now load the generated *bedgraph* files into *IGV* to visualize the results of the ChIP-seq experiment. We'll also load *the macs_peaks.narrowPeak* file to identify statistically significant peaks:



Analysis of FNR binding sequences

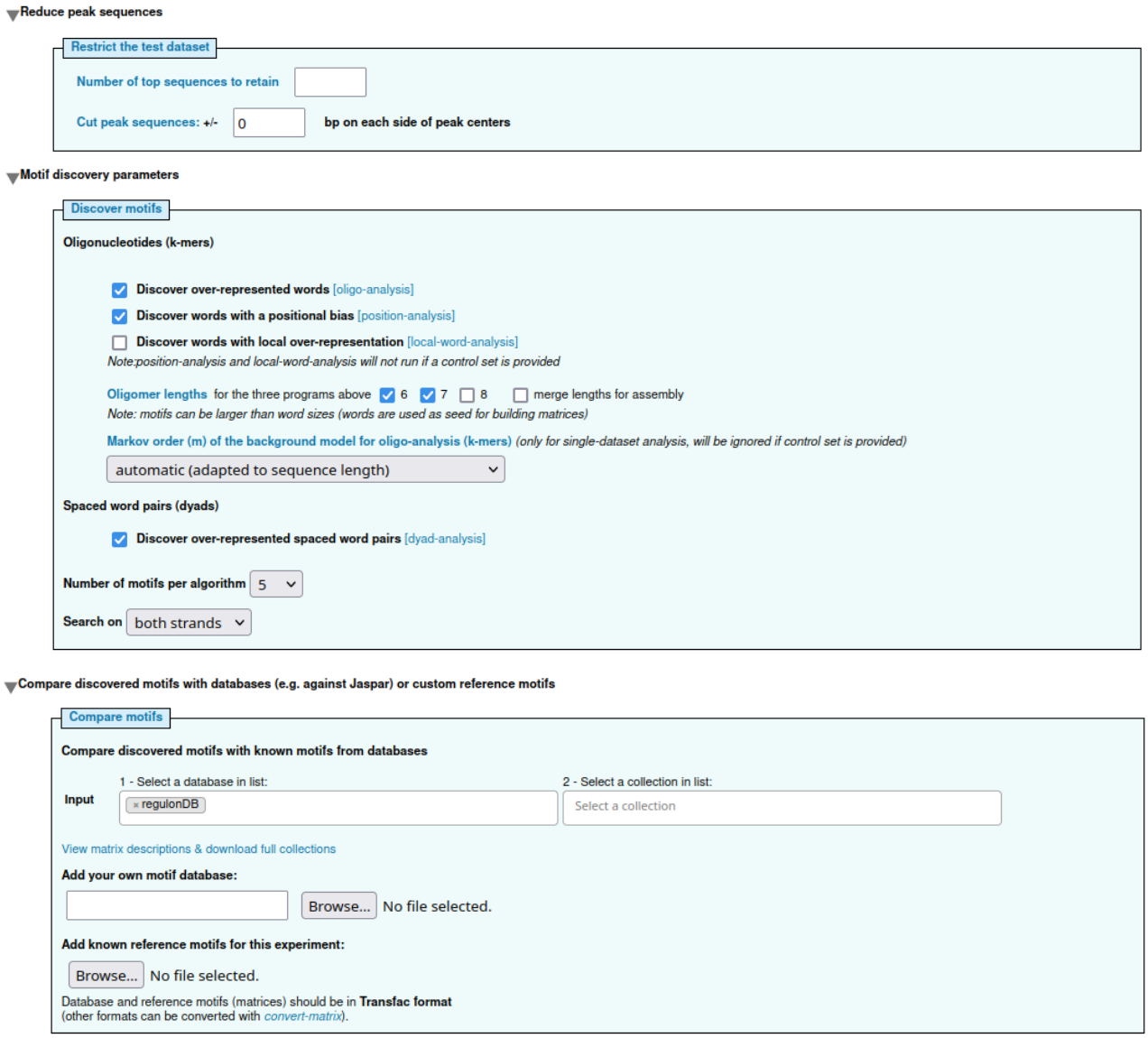
We already have the statistically significant peaks identified and the coordinates of each of them. Now we just need to recover the sequences of these to look for conserved motifs. We will do this with the *getfasta* program from the *bedtools* suite:

```
$ bedtools getfasta -fi reference/e_coli_genome.fa \
> -bed macs/macs_peaks.narrowPeak \
> -fo macs/macs_peaks.fa
```

Since some of the sequences are very long, we trimmed those longer than 200 bases with the Python script called *shorter_peaks.py* which can be found on the entry's GitHub. This program will trim the sequences leaving the top of each peak in the center. If the peak is not more than 100 bases from the end or beginning of the sequence, then it will not trim that side. The file *macs_shorter_peaks.fa* will be generated with the shortened sequences in *fasta* format.

The discovery of conserved motifs will be done with a web application from the *RSA-tools* suite called *peak-motifs*. This program, aside from discovering those short sequences conserved in DNA, also allows us to cross-reference the results with a database that contains the motifs to which known transcription factors bind. The website to use this program is as follows: http://rsat.sb-roscoff.fr/peak-motifs_form.cgi

Apart from adding the file *macs_shorter_peaks.fa*, we must make sure that the following options are activated:



The option *Discover over-represented spaced word pairs* allows us to detect conserved sequences of the same size at the ends, but variable inside them (the binding site of the transcription factor FNR that we study is of this type). It is very important that we select the *regulonDB* database and the *RegulonDB prokaryotes list (2015_08)*, which contains information on the binding sites of transcription factors in prokaryotes, in order to cross-reference the discovered motifs with the known motifs.

We click on GO and wait for the processing of the data. The results will show a list of the conserved motifs present in the sequences we entered and the comparison of these with those present in the *regulonDB* database. We can see that among the motifs discovered there are some related to the FNR transcription factor:

Motif 12 dyads_m1dyads_m1

k-mer sig=**23.96**; evalue=**1.1e-24**

regulonDB

name	id	strand	Nb overlap columns	% aligned	Pearson correlation	Normalized cor	aligned col. motif wwatTGAtctagaTCAAww	aligned col. match
FNR	FNR	D	14	0.7368	0.944	0.696	NA	NA
CRP	CRP	R	19	0.8636	0.732	0.632	NA	NA
Total matches= 2								

Conclusion

In this entry we have studied a basic workflow to find the most likely DNA binding sites of a transcription factor from the sequencing data of a ChIP-seq experiment. We have seen the importance of performing a correct quality control of both the sequencing and the ChIP-seq experiment, as well as knowing the most important parameters of the peak caller and the files it will return to us as output. We have also shown how to visualize the results of peak calling in *IGV* and retrieve the sequences of significant peaks. Finally, we used a web application to find the conserved motifs in the enriched sequences and cross-reference the information with that of a database containing the DNA sequences to which known transcription factors bind.

The code used in this article is on the blog's GitHub.