Genomic Computing Evaluation

Assignment 3: NGS Data Processing

Fabrizio Frasca

April 22, 2017

Introduction

```
mkdir homework
cd homework
ln -sn /home/lriva/lesson/homeworkFastqfiles/gata1.fastq .
ln -sn /home/lriva/lesson/homeworkFastqfiles/tal1.fastq .
ln -sn /home/lriva/lesson/homeworkFastqfiles/input.fastq .
ln -sn /home/lriva/public_html/lesson/chromsizes.tab .
```

These three fastq files contain the results of 2 ChIP-seq experiments and 1 control experiment. GATA1 and TAL1 are two mouse transcription factors, please check out the details for these samples here:

- http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM923575
- http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM923582
- http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM923585

The .sra raw data were downloaded and they were converted in fatstq files using the fastq-dump utility. We took 10 million reads out of those files, corresponding to 40 million lines of the fastq files

1 Checking the quality of the raw data

1.1

Use the /home/lriva/FastQC/fastqc tool to get a report for the quality for these three fastq files; write here below the commands that you would use for this purpose

It is sufficient to run fastqc with the .fastq files as input.

```
/home/lriva/FastQC/fastqc gata1.fastq
/home/lriva/FastQC/fastqc tal1.fastq
/home/lriva/FastQC/fastqc input.fastq
```

1.2

What is the length of the reads? Inspect the fastq files and the fastqc report

All the reads have length **41**. This can be captured by inspecting the *fastqc_data.txt* files in *gata1_fastqc*, *tal1_fastqc* and *input_fastqc* generated folders.

For all the three files we can read the following:

```
>>Sequence Length Distribution  pass
#Length Count
41      1.0E7
>>END_MODULE
```

meaning that the three starting fastq files have all the reads of length 41.

The .html fastqc reports cannot be studied because of the problems encountered with the ftp protocol.

2 Filtering the raw data

2.1

Use the fastx_artifacts_filter to remove artifacts

The commands are:

```
fastx_artifacts_filter -Q 33 -v -i gata1.fastq -o gata1_artifacts.fastq
fastx_artifacts_filter -Q 33 -v -i tal1.fastq -o tal1_artifacts.fastq
fastx_artifacts_filter -Q 33 -v -i input.fastq -o input_artifacts.fastq
```

where we have specified options -Q 33 to refer to the Illumina quality score scale and -i, -o, to specify the input and output files.

2.2

Use fastq_quality_trimmer to trim based with quality lower than 20, requiring a min final length of 30nt

We can use the following commands:

```
fastq_quality_trimmer -Q 33 -v -t 20 -l 30
-i gata1_artifacts.fastq -o gata1_artifacts_trim.fastq
fastq_quality_trimmer -Q 33 -v -t 20 -l 30
-i tal1_artifacts.fastq -o tal1_artifacts_trim.fastq
fastq_quality_trimmer -Q 33 -v -t 20 -l 30
-i input_artifacts.fastq -o input_artifacts_trim.fastq
```

where we used -t 20 to set the quality cutoff threshold to 20 and -l 30 to impose the minimum final length to 30 of the trimmed reads.

2.3

Use fastq_quality_filter to keep only those reads with al least 80% of high quality bases (quality > 20)

The commands are:

```
fastq_quality_filter -Q 33 -v -q 21 -p 80
-i gata1_artifacts_trim.fastq -o gata1_artifacts_trim_quality.fastq
fastq_quality_filter -Q 33 -v -q 21 -p 80
-i tal1_artifacts_trim.fastq -o tal1_artifacts_trim_quality.fastq
fastq_quality_filter -Q 33 -v -q 21 -p 80
-i input_artifacts_trim.fastq -o input_artifacts_trim_quality.fastq
```

where -p 80 is to enforce that at least 80% of the bases must be high quality, and -q 21 filters out all the reads with quality smaller than 21. The choice to go for 21 instead of 20 was made because -q option takes integers values and the requirement was for a quality *strictly* greater than 20.

3 Checking the quality of the filtered data

3.1

Use the /home/lriva/FastQC/fastqc tool to get a report for the quality for the filtered fastq files

The commands are, again:

```
/home/lriva/FastQC/fastqc gata1_artifacts_trim_quality.fastq
/home/lriva/FastQC/fastqc tal1_artifacts_trim_quality.fastq
/home/lriva/FastQC/fastqc input_artifacts_trim_quality.fastq
```

3.2

Compare the results of 3.1) with 1.1) and briefly describe differences

By comparing the generated files *fastqc_data.txt* for all the couples:

- input *vs* input_artifacts_trim_quality
- gata1 *vs* gata1_artifacts_trim_quality
- tal1 *vs* tal1_artifacts_trim_quality

it is possible to notice that, in general, all the filtering operations together significantly enhanced the average quality of the reads.

In particular, the quality on the 'tail' is generally higher and less dispersed around its mean value - smaller interquartile difference. This causes the average-quality trend to be more constant, and we can indeed notice there no more heavy drops like before.

Finally, also consider the 10th percentile on the tail is now generally much higher and the count for N bases on the starting and ending positions is greatly reduced, and this is because such bases are cleaved off by the trimming operation, being very poor quality bases.

3.3

Which sample got the more reads discarded? how many reads still remain for this sample?

Since the initial number of reads was 10 millions for all the three samples, it is sufficient to look for the sample which has the minimum number of lines in the filtered fastq file:

```
$ wc -l gata1_artifacts_trim_quality.fastq
38151468 gata1_artifacts_trim_quality.fastq

$ wc -l tal1_artifacts_trim_quality.fastq
38484388 tal1_artifacts_trim_quality.fastq

$ wc -l input_artifacts_trim_quality.fastq
38185936 input_artifacts_trim_quality.fastq
```

The sample with the more reads discarded is **gata1** and the the number of reads still remaining for that sample is $38,151,468 \div 4$, that is **9,537,867**.

4 Reads alignment

4.1

Align the reads to the mm9 mouse genome; for each of the tasks here below, write below the commands that you would use to perform the required task

4.1.1

Determine the .sai file containing the bwa index alignments using the "bwa align" command

```
bwa aln -t4 -f gata1.sai /db/bwa/0.6.2/mm9/mm9
gata1_artifacts_trim_quality.fastq
bwa aln -t4 -f tal1.sai /db/bwa/0.6.2/mm9/mm9
tal1_artifacts_trim_quality.fastq
bwa aln -t4 -f input.sai /db/bwa/0.6.2/mm9/mm9
input_artifacts_trim_quality.fastq
```

4.1.2

Convert these coordinates in genomic coordinates using the "bwa samse" command, export the results using "samtools view" and finally sort the aligned reads based on their genomic positions using "samtools sort"

It is convenient the three commands `samse`, `samtools view` and `samtools sort` in a pipelined fashion:

```
bwa samse /db/bwa/0.6.2/mm9/mm9 gata1.sai
      gata1_artifacts_trim_quality.fastq |
              samtools view -ut /db/bwa/0.6.2/mm9/mm9 - |
              samtools sort - gata1
bwa samse /db/bwa/0.6.2/mm9/mm9 tal1.sai
      tal1_artifacts_trim_quality.fastq |
              samtools view -ut /db/bwa/0.6.2/mm9/mm9 - |
              samtools sort - tal1
bwa samse /db/bwa/0.6.2/mm9/mm9 input.sai
      input_artifacts_trim_quality.fastq |
              samtools view -ut /db/bwa/0.6.2/mm9/mm9 - |
              samtools sort - input
```

4.1.3

Create an index files for quick access to the .bam file using the "samtools index" command

It is sufficient to call the command `samtools index` by passing as input the three .bam files obtained in the step before:

```
samtools index gata1.bam
samtools index tal1.bam
samtools index input.bam
```

4.2

What is the percentage of reads successfully aligned? use the "samtools view" command playing with the "-c" and "-F 4" options; answer for all three filtered fastq files, also writing the commands that you would use to perform the required task

The three commands below output the number of successfully mapped reads:

```
$ samtools view -F 4 -c gata1.bam
8606362
$ samtools view -F 4 -c tal1.bam
9292370
$ samtools view -F 4 -c input.bam
9303996
```

Then we can calculate the percentages of successfully mapped reads as follows:

$$gata1_{\%} = 100 \times \frac{mapped\ filtered\ reads}{overall\ filtered\ reads} = 100 \times \frac{8606362}{9537867} \approx 90.23$$

$$tal1_{\%} = 100 \times \frac{mapped\ filtered\ reads}{overall\ filtered\ reads} = 100 \times \frac{9292370}{9621097} \approx 96.58$$

$$input_{\%} = 100 \times \frac{mapped\ filtered\ reads}{overall\ filtered\ reads} = 100 \times \frac{9303996}{9546484} \approx 97.45$$

5 Peak calling

5.1

Use MACS to call ChIP-seq peaks; write here below the commands that you would use to perform the required task

In order to perform peak calling, macs14 is used with the two transcription factors obtained .bam files as treatment (`-t` parameter), and with the input obtained .bam file as control for both (`-c` parameter).

Here below the two commands:

```
macs14 -t gata1.bam -c input.bam -n gata1 -m 10,30
    -f BAM -p 1e-8 -g mm -s 41 --diag
macs14 -t tal1.bam -c input.bam -n tal1 -m 10,30
    -f BAM -p 1e-8 -g mm -s 41 --diag
```

Among all the other parameters: `-p` is used to set the Pvalue cutoff for peak detection at 1e-8, `-g` is to indicate the genome length is the one of the mouse, `-s` is used to set the tag size to 41 and `--diag` is to ask macs to perform saturation analysis for peaks in several enrichment-ratio intervals.

5.2

How many positive and negative peaks were identified? (where negative peaks, in MACS terms, could be intended here as False Positives)

The number of peaks can be retrieved by counting the number of lines in the *gata1_peaks.bed* and *tal1_peaks.bed* generated files:

```
$ wc -l gata1_peaks.bed
2603

$ wc -l tal1_peaks.bed
1553
```

The number of negative peaks can instead be computed by **subtracting** 1 to the number of lines in the *gata1_negative_peaks.xls* and the *tal1_negative_peaks.xls* generated files. This is because the .xls files contain a one-line header.

```
$ wc -l gata1_negative_peaks.xls
8

$ wc -l tal1_negative_peaks.xls
8
```

That is: both tal1 and gata1 have 7 negative peaks identified.

5.3

What is the overall expected False Discovery Rate?

The False Discovery Rate is defined as the expected value of the proportion of false discoveries among the discoveries: $FDR = \mathbb{E}[V/R]$ where V is the number of false positives and R is the number of discoveries.

MACS uses to compute the FDR for each peak and store such values at the ninth column of the ”peaks.xls” generated file. In order to compute the FDR value for each peak it does the following:

- fix the pvalue threshold as the peak pvalue
- call peaks according to the set pvalue
- obtain x peaks
- swap treatment and control and perform peak calling with the same pvalue
- obtain y peaks
- estimate the FDR as $\frac{y}{x}$

In order to compute an *overall* expected False Discovery Rate one choice can be to act accordingly, so that FDRs for gata1 and tal1 can be estimated, respectively, as:

$$FDR_{\text{gata1}} = \frac{\text{gata1 negative peaks}}{\text{gata1 peaks}} = \frac{7}{2603} \approx 0.0027$$

$$FDR_{\text{tal1}} = \frac{\text{tal1 negative peaks}}{\text{tal1 peaks}} = \frac{7}{1553} \approx 0.0045$$

5.4

Regarding the False Negatives, what are gata1 and tal1 saturation levels with 90% of the reads at an enrichment level of 20-40 fold?

To answer this question we have to inspect the *gata1_diag.xls* and *tal1_diag.xls* generated files. The value at the second row and third column refers to the coverage by a sampling of 90% for the peaks with an enrichment rate in the interval 20-40, so this is the value we are looking for. We have:

gata1: 90.28
tal1: 96.38

6 Identifying the binding events shared by the two transcription factors and those that are specific

6.1

Use the "intersectBed" command to identify how many peaks are shared between gata1 and tal1, based on the output of 5.1); write here below the commands that you would use to perform the required task and report the number of found peaks

The commands we can use in order to do this are:

```
$ bedtools intersect -a gata1_peaks.bed
      -b tal1_peaks.bed -wa > int.bed
$ wc -l int.bed
1228
```

The result is that we have 1228 peaks which are shared between the two treatments.

6.2

Check the documentation of the "intersectBed" command to identify how many peaks are specific for GATA1; write here below the commands that you would use to perform the required task and report the number of found peaks

The option `-v` can be used to extract regions of the "`A`" sample which do not overlap with any region of the "`B`" sample. From the manual:

-v Only report those entries in A that have no overlap in B. Restricted by -f and -r.

Thus, we can use the following commands:

```
$ bedtools intersect -a gata1_peaks.bed
      -b tal1_peaks.bed -v > int_gata1.bed
$ wc -l int_gata1.bed
1377
```

7 Displaying the results in the genome browser

7.1

Generate the bw files to be exported in the genome browser; for each of the subtask here below, write below the commands that you would use to perform the required task

7.1.1

Convert the bam files in bed files using the "bamToBed" command

Let's first remove possible duplicates:

```
samtools rmdup -s gata1.bam gata1.nodup.bam
samtools rmdup -s tal1.bam tal1.nodup.bam
samtools rmdup -s input.bam input.nodup.bam
```

Then we can call the command of interest:

```
bamToBed -i gata1.nodup.bam > gata1.bed
bamToBed -i tal1.nodup.bam > tal1.bed
bamToBed -i input.nodup.bam > input.bed
```

7.1.2

Create in silico extension of the peaks coordinates adding 160 bases to the right side in a strand specific manner, use the "slopBed" command

The commands are:

```
bedtools slop -i gata1.bed -g chromsizes.tab -s -l 0 -r 160
  > gata1_slop.bed
bedtools slop -i tal1.bed -g chromsizes.tab -s -l 0 -r 160
  > tal1_slop.bed
bedtools slop -i input.bed -g chromsizes.tab -s -l 0 -r 160
  > input_slop.bed
```

7.1.3

Determine the count of reads for each base in the genome using the "genomeCoverageBed" command

In order to determine the count *for each base*, we should use the `-d` option:

```
genomeCoverageBed -i gata1_slop.bed -g chromsizes.tab -d
  > gata1_perbase.wiggle
genomeCoverageBed -i tal1_slop.bed -g chromsizes.tab -d
  > tal1_perbase.wiggle
genomeCoverageBed -i input_slop.bed -g chromsizes.tab -d
  > input_perbase.wiggle
```

However, if we want to upload the files on the UCSC genome browser, a more suitable option is to generate the output in bedgraph format, in this way:

```
genomeCoverageBed -i gata1_slop.bed -g chromsizes.tab -bg
  > gata1.wiggle
genomeCoverageBed -i tal1_slop.bed -g chromsizes.tab -bg
  > tal1.wiggle
genomeCoverageBed -i input_slop.bed -g chromsizes.tab -bg
  > input.wiggle
```

7.1.4

Convert the .wg file in its binary bigwig version using the "wigToBigWig" command

It is sufficient to use the commands:

```
wigToBigWig gata1.wiggle chromsizes.tab gata1.bw
wigToBigWig tal1.wiggle chromsizes.tab tal1.bw
wigToBigWig input.wiggle chromsizes.tab input.bw
```

7.1.5

Go to http://genome.ucsc.edu/ and create three custom tracks with the three .bw files created in 7.1.4) and provide screen-shots of the uploaded tracks

The tracks were uploaded by following `My data > Custom Tracks`, by selecting

clade:	Mammal
genome:	Mouse
assembly:	July 2007 (NCBI37/mm9)

and, finally, by providing the following lines to be submitted:

```
track type=bigWig name="gata1"
      bigDataUrl=http://bioserver.iit.ieo.eu/~lriva/homeworkbw/gata1.bw

track type=bigWig name="tal1"
      bigDataUrl=http://bioserver.iit.ieo.eu/~lriva/homeworkbw/tal1.bw

track type=bigWig name="input"
      bigDataUrl=http://bioserver.iit.ieo.eu/~lriva/homeworkbw/input.bw
```

Here is a screenshot of the successfully updated tracks:

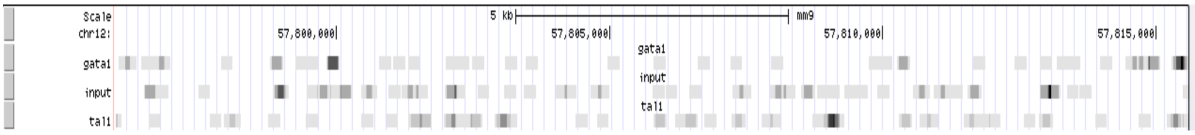


Figure 1: gata1, tal1 and input tracks uploaded on the UCSC genome browser

7.1.6

Based on 6.1) and 6.2) identify the genomic coordinates of one shared peak and of one peak specific for GATA1; browse to those locations and generate a plot in the genome browser reporting these two regions, and, finally, provide the screen-shot of the created plot

One shared peak is for the genomic coordinates `chrX:53,981,724-53,982,059` and can be found in the `int.bed` generated file. Figures 2 and 3 depict the peaks in the UCSC genome browser.

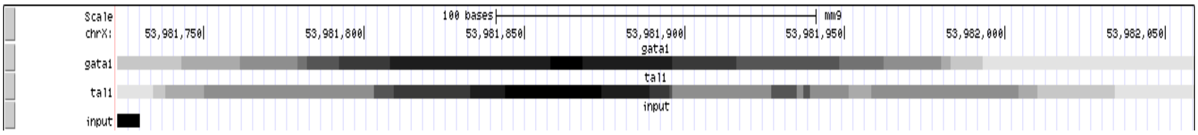


Figure 2: shared peak for the coordinates: chrX:53,981,724-53,982,059, dense visualization

One peak specific for gata1 transcription factor is, instead, for the genomic coordinates `chrX:131,741,773-131,742,135` and can be found in the `int_gata1.bed` generated file. Figures 4 and 5 depict the peak in the UCSC genome browser.

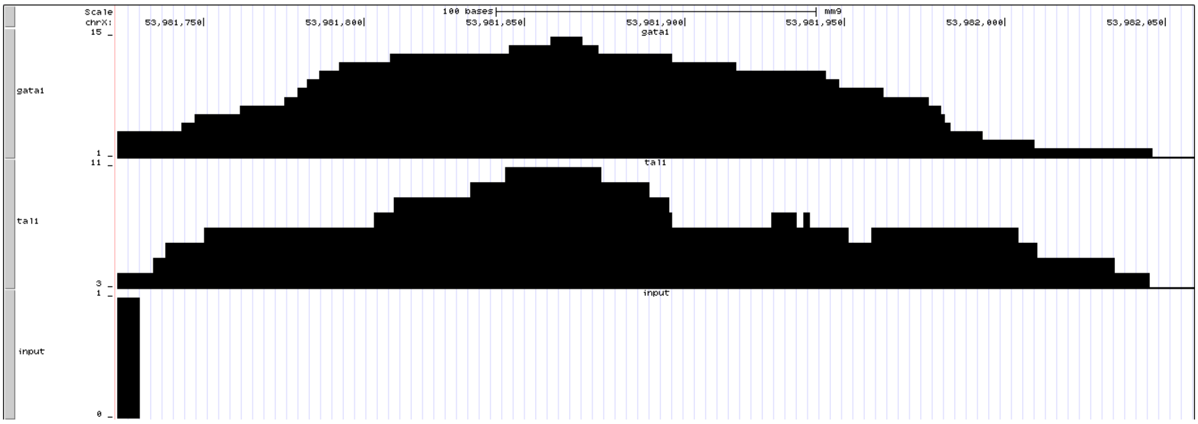


Figure 3: shared peak for the coordinates: chrX:53,981,724-53,982,059, full visualization

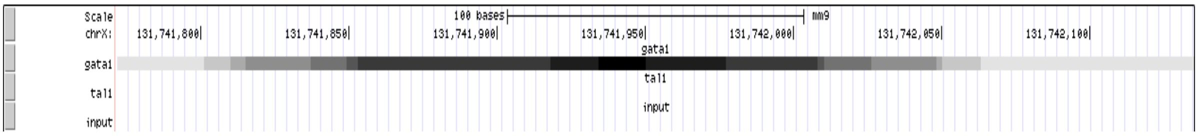


Figure 4: specific peak for the coordinates: chrX:131,741,773-131,742,135, dense visualization

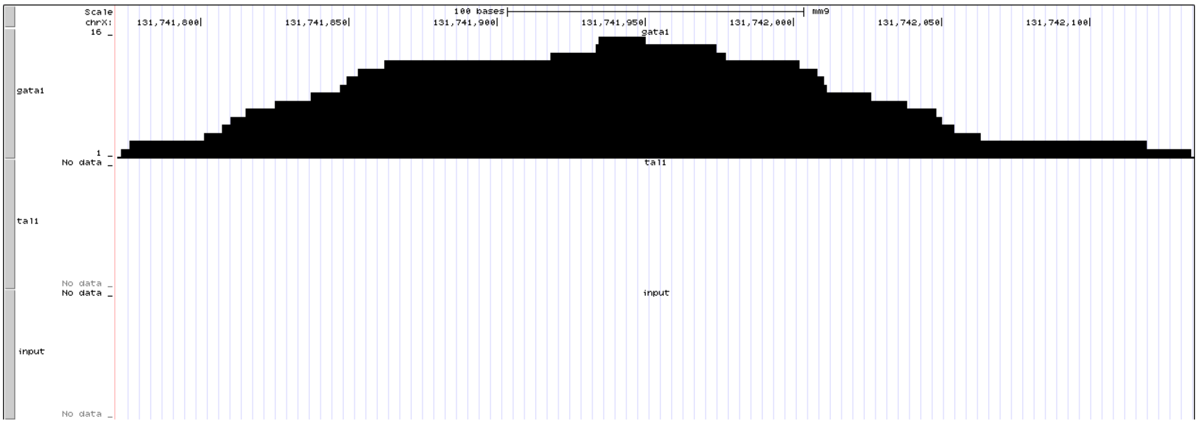


Figure 5: specific peak for the coordinates: chrX:131,741,773-131,742,135, full visualization

8 Peaks annotation and motif finding with GREAT

8.1

Use the "cut" linux command (with the -f argument) to generate the GREAT input files taking the first three columns of the .bed MACS peak files generated at 6.1) and 6.2)

In order to take the first three columns only, it is sufficient to use the `-f 1-3` option with the cut command.

```
cut -f 1-3 int.bed > int_great.bed
cut -f 1-3 spec_gata1.bed > spec_gata1_great.bed
```

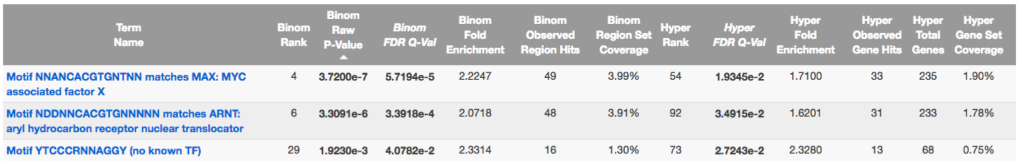
8.2

Go to great.stanford.edu and upload the files obtained in 8.1) matching it to the mouse mm9 genome. Compare the motifs identified for the two peak sets; provide their screen shots and comment them briefly

As for the *int_great.bed* file, the reported results (figure 6) correspond to the promoter motifs where a **joint** gata1 and tal1 binding is observed.

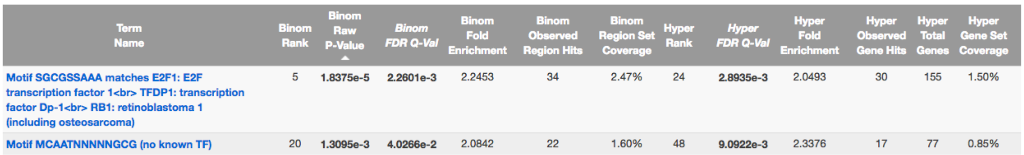
As for the *spec_gata_great.bed* file, the reported results (figure 7) correspond to the promoter motifs where a **specific** gata1 binding is observed.

The first motifs for both the two GREAT jobs have quite good pvalues.



Term Name	Binom Rank	Binom Raw P-Value ▲	*Binom FDR Q-Val*	Binom Fold Enrichment	Binom Observed Region Hits	Binom Region Set Coverage	Hyper Rank	*Hyper FDR Q-Val*	Hyper Fold Enrichment	Hyper Observed Gene Hits	Hyper Total Genes	Hyper Gene Set Coverage
Motif NNANCACGTGNTNN matches MAX: MYC associated factor X	4	**3.7200e-7**	**5.7194e-5**	2.2247	49	3.99%	54	**1.9345e-2**	1.7100	33	235	1.90%
Motif NDDNNCACGTGNNNNN matches ARNT: aryl hydrocarbon receptor nuclear translocator	6	**3.3091e-6**	**3.3918e-4**	2.0718	48	3.91%	92	**3.4915e-2**	1.6201	31	233	1.78%
Motif YTCCCRNNAGGY (no known TF)	29	**1.9230e-3**	**4.0782e-2**	2.3314	16	1.30%	73	**2.7243e-2**	2.3280	13	68	0.75%

Figure 6: Predicted promoter motifs for the regions where both gata1 and tal1 transcription factors bind.



Term Name	Binom Rank	Binom Raw P-Value ▲	*Binom FDR Q-Val*	Binom Fold Enrichment	Binom Observed Region Hits	Binom Region Set Coverage	Hyper Rank	*Hyper FDR Q-Val*	Hyper Fold Enrichment	Hyper Observed Gene Hits	Hyper Total Genes	Hyper Gene Set Coverage
Motif SGCGSSAAA matches E2F1: E2F transcription factor 1 TFDP1: transcription factor Dp-1 RB1: retinoblastoma 1 (including osteosarcoma)	5	**1.8375e-5**	**2.2601e-3**	2.2453	34	2.47%	24	**2.8935e-3**	2.0493	30	155	1.50%
Motif MCAATNNNNNGCG (no known TF)	20	**1.3095e-3**	**4.0266e-2**	2.0842	22	1.60%	48	**9.0922e-3**	2.3376	17	77	0.85%

Figure 7: Predicted promoter motifs for the regions where only gata1 transcription factors binds.