

Genomic Computing Evaluation

Assignment 3: NGS Data Processing

Fabrizio Frasca

April 9, 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 fastq 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 understood by inspecting...

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 at 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 tal1_artifacts_trim_quality and input vs tal1_artifacts_trim_quality, it is possible to notice that, in general, all the filtering operations together greatly enhanced the average quality of the reads. In particular:

- Each base position has a greater quality, in average
- The quality on the 'tail' is higher and so the average-quality trend tends to be more constant, i.e. there are no drops anymore
- The values on the tail are less dispersed around the mean, i.e. the interquartile difference is smaller
- The 10th percentile on the tail is now much higher than before.
- The count for N bases on the starting and ending positions is greatly reduced, as 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:

$$gata1\% = 100 \times \frac{\text{mapped filtered reads}}{\text{overall filtered reads}} = 100 \times \frac{8606362}{9537867} \approx 90.23$$

$$tal1\% = 100 \times \frac{\text{mapped filtered reads}}{\text{overall filtered reads}} = 100 \times \frac{9292370}{9621097} \approx 96.58$$

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

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 positive and negative peaks can be retrieved by the number of lines in the ... generated files:

We have ...

5.3

What is the overall expected False Discovery Rate?

...

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?

We can inspect the ... and ... generated files for the value corresponding to the second row () and third column (). 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
```

7 Displaying the results in the genome browser

8 Peaks annotation and motif finding with GREAT