Genomic Regulation

Ivó Hernández, Helena Liz, Diego Fuentes, Fernando Freire

March 3, 2019

Contents

1 Genomic Regulation

1.1 Get chr16 CTCF segments

Get the *chr16* segments which share the same state between both monocyte replicates.

1.1.1 Test files

Script 1.1.1 (text)

```bash
%%bash
# Obtain files for test
cd RESULTS/Modelo_11_estados
cat Monocyte1_11_Master_11_segments.bed | grep 'chr16' | grep 'E9' | sort -k 2,3 -h | head
    -n 20 > monocyte1_segments.bed
cat Monocyte2_11_Master_11_segments.bed | grep 'chr16' | grep 'E9' | sort -k 2,3 -h | head
    -n 20 > monocyte2_segments.bed
echo "Monocyte 1 segments.bed"
cat monocyte1_segments.bed
wc -l monocyte1_segments.bed
echo "Monocyte 2 segments.bed"
cat monocyte2_segments.bed
wc -l monocyte2_segments.bed
```

Output

```
Monocyte 1 segments.bed
chr16       60400       61400       E9
chr16       72600       72800       E9
chr16       115200      116000      E9
chr16       146400      147400      E9
chr16       156600      157600      E9
chr16       167800      168200      E9
chr16       232200      232400      E9
chr16       412000      412600      E9
chr16       441800      442200      E9
chr16       486400      486800      E9
chr16       537600      538000      E9
chr16       597000      597600      E9
chr16       629000      629400      E9
chr16       661000      661600      E9
chr16       710800      711200      E9
chr16       711600      711800      E9
chr16       736200      736400      E9
chr16       761200      763200      E9
chr16       835400      836200      E9
chr16       1019400     1019600     E9
      20 monocyte1_segments.bed
Monocyte 2 segments.bed
chr16       60400       61400       E9
chr16       72400       72800       E9
chr16       115000      116400      E9
```

```
chr16          146600          147400          E9
chr16          155400          158200          E9
chr16          167800          168800          E9
chr16          231800          232200          E9
chr16          309000          309200          E9
chr16          353600          354200          E9
chr16          402200          403200          E9
chr16          412000          412800          E9
chr16          441800          442200          E9
chr16          508200          508400          E9
chr16          537400          538200          E9
chr16          596600          597400          E9
chr16          627800          630600          E9
chr16          660800          661800          E9
chr16          710800          711800          E9
chr16          717400          718200          E9
chr16          735800          736800          E9
       20 monocyte2_segments.bed
```

Script 1.1.2 (text)

```
1   %%writefile bed1.bed
2   chr16          60400          61400          E9
3   chr16          72600          72800          E9
4   chr16          115200          116000          E9
5   chr16          146400          147400          E9
6   chr16          156600          157600          E9
7   chr16          167800          168200          E9
8   chr16          232200          232400          E9
9   chr16          412000          412600          E9
10  chr16          441800          442200          E9
11  chr16          486400          486800          E9
12  chr16          537600          538000          E9
13  chr16          597000          597600          E9
14  chr16          629000          629400          E9
15  chr16          661000          661600          E9
16  chr16          710800          711200          E9
17  chr16          711600          711800          E9
18  chr16          736200          736400          E9
19  chr16          761200          763200          E9
20  chr16          835400          836200          E9
21  chr16          1019400          1019600          E9
```

Output

```
Overwriting bed1.bed
```

Script 1.1.3 (text)

```
 1  %%writefile bed2.bed
 2  chr16      60400       61400       E9
 3  chr16      72400       72800       E9
 4  chr16      115000      116400      E9
 5  chr16      146600      147400      E9
 6  chr16      146610      147400      E8
 7  chr16      155400      158200      E9
 8  chr16      167800      168800      E9
 9  chr16      231800      232200      E9
10  chr16      309000      309200      E9
11  chr16      353600      354200      E9
12  chr16      402200      403200      E9
13  chr16      412000      412800      E9
14  chr16      441800      442200      E9
15  chr16      508200      508400      E9
16  chr16      537400      538200      E9
17  chr16      596600      597400      E9
18  chr16      596700      597400      E8
19  chr16      596700      597400      E8
20  chr16      627800      630600      E9
21  chr16      660800      661800      E9
22  chr16      710800      711800      E9
23  chr16      717400      718200      E9
24  chr16      735800      736800      E9
```

Output

```
Overwriting bed2.bed
```

Script 1.1.4 (text)

```
 1  %%writefile dnase1.peaks.bed
 2  chr1       770942      771278      chr1.9      584     .     0.039      1.79 ⌟
    ↪          -1          151
 3  chr1       771678      771933      chr1.10     568     .     0.0343     1.5 ⌟
    ↪  6       -1          121
 4  chr1       773279      773398      chr1.11     555     .     0.0303     1.3 ⌟
    ↪  8       -1          49
 5  chr1       777497      777598      chr1.12     553     .     0.0299     1.3 ⌟
    ↪  6       -1          46
 6  chr1       794051      794336      chr1.13     569     .     0.0344     1.5 ⌟
    ↪  7       -1          152
 7  chr1       800514      800667      chr1.14     549     .     0.0287
    ↪  1.3     -1          34
 8  chr1       805004      805656      chr1.15     1000    .     0.3561     16 ⌟
    ↪          -1          286
 9  chr16      63392       63462       chr16.1     551     .     0.0292     1.33 ⌟
    ↪          -1          26
```

```
10 chr16          65192          65500      chr16.2       650         .        0.0582      2.69 ⌋
   ↪              -1         140
11 chr16          65680          65848      chr16.3       578         .        0.0371
   ↪  1.7        -1          81
12 chr16          66552          66704      chr16.4       565         .        0.0334      1.52 ⌋
   ↪              -1          71
13 chr16          69567          69918      chr16.5       593         .        0.0416      1.91 ⌋
   ↪              -1         156
14 chr16          72620          73427      chr16.6      1000         .        0.2652      12. ⌋
   ↪  5           -1         256
15 chr16          74047          74486      chr16.7       687         .        0.069
   ↪  3.2         -1         213
16 chr16          77159          77214      chr16.8       550         .        0.0289      1.31 ⌋
   ↪              -1          20
17 chr16          78558          80270      chr16.9       756         .        0.0889      4.15 ⌋
   ↪              -1        1188
18 chr16          80539          84109      chr16.10      797         .        0.101       4.72 ⌋
   ↪              -1        2835
19 chr16          84632          86116      chr16.11     1000         .        0.2262      10 ⌋
   ↪  .7          -1         648
20 chr16          86223          86873      chr16.12      717         .        0.0777      3.6 ⌋
   ↪  2           -1         274
21 chr16          87530          87809      chr16.13      614         .        0.0475      2.1 ⌋
   ↪  9           -1         147
22 chr16          88254          89597      chr16.14      703         .        0.0736      3.4 ⌋
   ↪  2           -1         691
23 chr16          91933          92367      chr16.15      558         .        0.0312      1.4 ⌋
   ↪  2           -1          78
24 chr16         102933         104351      chr16.16     1000         .        0.521       1 ⌋
   ↪  6           -1         852
25 chr16         107650         109038      chr16.17      708         .        0.0751      3 ⌋
   ↪  .49         -1         327
26 chr16         109773         109913      chr16.18      555         .        0.0305      1 ⌋
   ↪  .39         -1          61
27 chr16         110337         110607      chr16.19      581         .        0.0381      1 ⌋
   ↪  .74         -1         117
28 chr16         110941         114948      chr16.20      915         .        0.1354      6 ⌋
   ↪  .36         -1         421
29 chr16         115448         116225      chr16.21     1000         .        0.3633      ⌋
   ↪  16          -1         419
30 chr16         116533         116836      chr16.22      568         .        0.0343      1 ⌋
   ↪  .56         -1         131
```

Output

```
Writing dnase1.peaks.bed
```

1.1.2 Methods

```python
def head(filename, lines=20):
    """
    """
    i = 0
    file = open(filename, "r")
    for line in file:
        print(line.strip())
        i += 1
        if i > lines:
            break
    file.close()

def get_parts(bed_line, sep='\t'):
    """
    """
    bed_line_parts = bed_line.rstrip('\n').split(sep)
    return bed_line_parts[0], int(bed_line_parts[1]), int(bed_line_parts[2]),
    ↪  bed_line_parts[3]

def concat_parts(chrom, start, end, feature, sep='\t'):
    """
    """
    bed_line = chrom + '\t' + str(start) +  '\t' + str(end) + '\t' + feature + '\n'
    return bed_line

def bed_coverage(filename, sep='\t'):
    """
    """
    i = 0
    file = open(filename, "r")
    coverage = 0
    for line in file:
        _ , f1_segment_start, f1_segment_end, _ = get_parts(line)
        coverage += f1_segment_end + f1_segment_start
    file.close()
    return coverage

def intersect_bed(input_dir, input_file1, input_file2, output_dir, output_file, chrom="chr16⏌
",
                  f1_feature_filter="E9", f2_feature_filter="E9", output_feature="E9",
                  ↪  sep='\t',
                  drop_feature_threshold=20, output_mode="intersect"):
    """
    If output node is intersect, returns the intersected bed segments
    If output mode is annotate, returns all the segments of input_file1
    annotated if it's the case with the feature defined in input_file2.
    """
    f1_segments = open(input_dir + "/" + input_file1, "r")
    f2_segments = open(input_dir + "/" + input_file2, "r")
```

```python
47    output_segments = open(output_dir + "/" + output_file, "w")
48    f1_segment = f1_segments.readline()
49    f2_segment = f2_segments.readline()
50    while(f1_segment != "" and f2_segment != ""):
51        f1_chrom, f1_segment_start, f1_segment_end, f1_feature = get_parts(f1_segment)
52        f2_chrom, f2_segment_start, f2_segment_end, f2_feature = get_parts(f2_segment)
53        feature = f1_feature
54        # Filter f1 and read f1
55        if f1_chrom != chrom or (f1_feature_filter != "" and f1_feature != f1_feature_filter⌋
    ):
56            f1_segment = f1_segments.readline()
57        # Filter f2 and read f2
58        elif f2_chrom != chrom or (f2_feature_filter != "" and f2_feature !=
        ↪  f2_feature_filter):
59            f2_segment = f2_segments.readline()
60        # f2 segment downstream f1 segment
61        elif f2_segment_start > f1_segment_end:
62            if output_mode == "annotate" and drop_feature_threshold < f1_segment_end -
            ↪  f1_segment_start:
63                output_segment = concat_parts(chrom, f1_segment_start, f1_segment_end,
                ↪  feature)
64                output_segments.write(output_segment)
65            f1_segment = f1_segments.readline()
66        # f1 segment downstream f2 segment
67        elif f1_segment_start > f2_segment_end:
68            f2_segment = f2_segments.readline()
69        else: # Intersect
70            # Save intersect
71            if output_mode == "intersect":
72                output_start = max(f1_segment_start, f2_segment_start)
73                output_end =  min(f2_segment_end, f1_segment_end)
74                if drop_feature_threshold < output_end - output_start:
75                    output_segment = concat_parts(chrom, output_start, output_end,
                    ↪  output_feature)
76                    output_segments.write(output_segment)
77            else:
78                feature = f1_feature + "+" + output_feature
79            # Advance f1
80            if f2_segment_end >= f1_segment_end:
81                if output_mode == "annotate" and drop_feature_threshold < f1_segment_end -
                ↪  f1_segment_start:
82                    output_segment = concat_parts(chrom, f1_segment_start, f1_segment_end,
                    ↪  feature)
83                    output_segments.write(output_segment)
84                f1_segment = f1_segments.readline()
85            # Advance f2
86            elif f1_segment_end > f2_segment_end:
87                f2_segment = f2_segments.readline()
88    while(output_mode == "annotate" and f1_segment != ""):
89        output_segments.write(f1_segment)
90        f1_segment = f1_segments.readline()
91
```

```
92        f1_segments.close()
93        f2_segments.close()
94        output_segments.close()
```

1.1.3 Tests

Script 1.1.6 (python)

```python
1  PATH = "RESULTS/Modelo_11_estados"
2  M1_FILE = "Monocyte1_11_Master_11_segments.bed"
3  M2_FILE = "Monocyte2_11_Master_11_segments.bed"
4  M1_FILE_TEST = "bed1.bed"
5  M2_FILE_TEST = "bed2.bed"
6  SEP = '\t'
7  CHR = "chr16"
8  STATE = "E9"
9
10 intersect_bed(".", M1_FILE_TEST, M2_FILE_TEST, ".", "E9_segments_test.bed", chrom=CHR,
11                   f1_feature_filter=STATE, f2_feature_filter=STATE, output_feature=STATE,
                     ↪  sep=SEP,
12                   drop_feature_threshold=300)
13
14 head("E9_segments_test.bed", 20)
```

Output

```
here
chr16         60400         61400         E9
chr16        115200        116000        E9
chr16        146600        147400        E9
chr16        156600        157600        E9
chr16        167800        168200        E9
chr16        412000        412600        E9
chr16        441800        442200        E9
chr16        537600        538000        E9
chr16        597000        597400        E9
chr16        629000        629400        E9
chr16        661000        661600        E9
chr16        710800        711200        E9
```

1.1.4 Goal

Script 1.1.7 (python)

```python
1  intersect_bed(PATH, M1_FILE, M2_FILE, ".", "E9_segments.bed", chrom="chr16",
2                    f1_feature_filter="E9", f2_feature_filter="E9", output_feature="E9",
                     ↪  sep=SEP,
3                    drop_feature_threshold=20)
```

```
4
5  head("E9_segments.bed", 10)
```

Output

```
chr16        60400        61400        E9
chr16        72600        72800        E9
chr16        115200       116000       E9
chr16        146600       147400       E9
chr16        156600       157600       E9
chr16        167800       168200       E9
chr16        412000       412600       E9
chr16        441800       442200       E9
chr16        537600       538000       E9
chr16        597000       597400       E9
chr16        629000       629400       E9
```

1.2 Segment annotation

Anotar los segmentos. Como mínimo, se deberá dar el porcentaje de segmentos que solapan con protein-coding genes en dicho cromosoma.

1.3 DNASE I overlap

Download the peaks of DNase I in monocytes of ENCODE for chr16 and calculate the percentage of overlap between DNaseI-peaks and your work segments. Use the file wgEncodeOpenChromDnaseMonocd14Pk.narrowPeak.gz in: http://hgdownload.cse.ucsc.edu/goldenpath/hg19/encodeDCC/wgEncodeOpenChrom/Dnase

1.3.1 Tests

Script 1.3.1 (python)

```
1  intersect_bed(".", "E9_segments_test.bed", "dnase1.peaks.bed", ".",
   ↪  "E9_dnase1_overlap_test.bed", chrom="chr16",
2                f1_feature_filter="E9", f2_feature_filter="",
                 ↪  output_feature="E9_dnase1_overlap", sep=SEP,
3                drop_feature_threshold=20)
4  head("E9_dnase1_overlap_test.bed", 10)
```

Output

```
here
chr16        115448       116000       E9_dnase1_overlap
```

Script 1.3.2 (python)

```python
intersect_bed(".", "E9_segments.bed", "dnase1.peaks.bed", ".", "E9_dnase1_overlap_test.bed",
 ↪  chrom="chr16",
                f1_feature_filter="E9", f2_feature_filter="",
                 ↪  output_feature="E9_dnase1_overlap", sep=SEP,
                drop_feature_threshold=20)
head("E9_dnase1_overlap_test.bed", 10)
```

Output

```
chr16        72620        72800        E9_dnase1_overlap
chr16        115448       116000          E9_dnase1_overlap
```

1.3.2 Overlap

Script 1.3.3 (python)

```python
intersect_bed(".", "E9_segments.bed", "wgEncodeOpenChromDnaseMonocd14Pk.narrowPeak", ".",
                "E9_dnase1_overlap.bed", chrom="chr16",
                f1_feature_filter="E9", f2_feature_filter="",
                 ↪  output_feature="E9_dnase1_overlap", sep=SEP,
                drop_feature_threshold=20)
head("E9_dnase1_overlap.bed", 10)
```

Output

```
chr16        72620        72800        E9_dnase1_overlap
chr16        115448       116000          E9_dnase1_overlap
chr16        146819       147400          E9_dnase1_overlap
chr16        157056       157367          E9_dnase1_overlap
chr16        167800       168118          E9_dnase1_overlap
chr16        412000       412600          E9_dnase1_overlap
chr16        441800       442200          E9_dnase1_overlap
chr16        537761       538000          E9_dnase1_overlap
chr16        597000       597400          E9_dnase1_overlap
chr16        629000       629400          E9_dnase1_overlap
chr16        661000       661548          E9_dnase1_overlap
```

Script 1.3.4 (python)

```python
intersect_bed(".", "E9_segments.bed", "wgEncodeOpenChromDnaseMonocd14Pk.narrowPeak", ".",
                "E9_dnase1_overlap_annotate.bed", chrom="chr16",
                f1_feature_filter="E9", f2_feature_filter="",
                 ↪  output_feature="E9_dnase1_overlap", sep=SEP,
                drop_feature_threshold=20, output_mode="annotate")
head("E9_dnase1_overlap_annotate.bed", 10)
```

```
here
chr16          60400          61400          E9
chr16          72600          72800          E9+E9_dnase1_overlap
chr16          115200         116000         E9+E9_dnase1_overlap
chr16          146600         147400         E9+E9_dnase1_overlap
chr16          156600         157600         E9
chr16          167800         168200         E9
chr16          412000         412600         E9+E9_dnase1_overlap
chr16          441800         442200         E9+E9_dnase1_overlap
chr16          537600         538000         E9+E9_dnase1_overlap
chr16          597000         597400         E9+E9_dnase1_overlap
chr16          629000         629400         E9+E9_dnase1_overlap
```

Script 1.3.5 (text)

```bash
%%bash
# Merge
wc -l E9_dnase1_overlap_annotate.bed
wc -l E9_segments.bed
tail E9_dnase1_overlap_annotate.bed
echo
tail E9_segments.bed
echo
head E9_dnase1_overlap_annotate.bed
echo
head E9_segments.bed
```

Output

```
     468 E9_dnase1_overlap_annotate.bed
     468 E9_segments.bed
chr16          89233600       89234800       E9
chr16          89527000       89527400       E9
chr16          89623800       89624200       E9+E9_dnase1_overlap
chr16          89707800       89708000       E9+E9_dnase1_overlap
chr16          89772400       89772600       E9+E9_dnase1_overlap
chr16          89927000       89927800       E9
chr16          89976600       89977000       E9+E9_dnase1_overlap
chr16          90092400       90092800       E9+E9_dnase1_overlap
chr16          90182400       90183000       E9
chr16          90281600       90282000       E9

chr16          89233600       89234800       E9
chr16          89527000       89527400       E9
chr16          89623800       89624200       E9
chr16          89707800       89708000       E9
chr16          89772400       89772600       E9
chr16          89927000       89927800       E9
chr16          89976600       89977000       E9
```

```
chr16          90092400          90092800          E9
chr16          90182400          90183000          E9
chr16          90281600          90282000          E9

chr16          60400          61400          E9
chr16          72600          72800          E9+E9_dnase1_overlap
chr16          115200          116000          E9+E9_dnase1_overlap
chr16          146600          147400          E9+E9_dnase1_overlap
chr16          156600          157600          E9
chr16          167800          168200          E9
chr16          412000          412600          E9+E9_dnase1_overlap
chr16          441800          442200          E9+E9_dnase1_overlap
chr16          537600          538000          E9+E9_dnase1_overlap
chr16          597000          597400          E9+E9_dnase1_overlap

chr16          60400          61400          E9
chr16          72600          72800          E9
chr16          115200          116000          E9
chr16          146600          147400          E9
chr16          156600          157600          E9
chr16          167800          168200          E9
chr16          412000          412600          E9
chr16          441800          442200          E9
chr16          537600          538000          E9
chr16          597000          597400          E9
```

Script 1.3.6 (python)

```python
1  coverage_peaks = bed_coverage("wgEncodeOpenChromDnaseMonocd14Pk.narrowPeak", sep='\t')
2  coverage_E9 = bed_coverage("E9_segments.bed", sep='\t')
3
4  print("Coverage DNASE peaks", coverage_peaks)
5  print("Coverage E9", coverage_E9)
6  print("Percent overlap over total coverage peaks:", coverage_E9 * 100 / coverage_peaks)
```

Output

```
Coverage DNASE peaks 22653113851288
Coverage E9 43366167200
Percent overlap over total coverage peaks: 0.19143578884866774
```