4. EN-TEx ATAC-seq data: downstream analyses

Move to folder ATAC-seq, and create folders to store bigBed data files and peaks analyses files. Make sure the files are organized in a consistent way as done for ChIP-seq.

- Create folders

```
mkdir data
mkdir analyses
mkdir data
```

Retrieve from a newly generated metadata file ATAC-seq peaks (bigBed narrow, pseudoreplicated peaks, assembly GRCh38) for stomach and sigmoid_colon for the same donor used in the previous sections. Hint: have a look at what we did here. Make sure your md5sum values coincide with the ones provided by ENCODE.

../bin/download.metadata.sh
https://www.encodeproject.org/metadata/?replicates.library.biosample.donor.uuid=d370683e-81e7-473f-8475-7716d027849b&status=released&status=submitted&status=in+progress&assay_slims=DNA+accessibility&biosample_ontology.term_name=sigmoid+colon&biosample_ontology.term_name=stomach&assay_title=ATAC-seq&type=Experiment

- Download peak calling and fold-change signal files

```
mkdir data/bigBed.files data/bigWig.files
```

- Obtain the corresponding IDs

```
grep -F "bigBed_narrowPeak" metadata.tsv |\
grep -F "pseudoreplicated_peaks" |\
grep -F "GRCh38" |\
awk 'BEGIN{FS=OFS="\t"}{print $1, $11, $23}' |\
sort -k2,2 -k1,1r |\
sort -k2,2 -u > analyses/bigBed.peaks.ids.txt
```

- Obtain the bigBed files for stomach and sigmoid colon using the IDs obtained before.

```
cut -f1 analyses/bigBed.peaks.ids.txt |\
while read filename; do
  wget -P data/bigBed.files
"https://www.encodeproject.org/files/$filename/@@download/$filename.bigBed"
done
```

- Check integrity of files by verifying their MD5 hash, a sort of digital fingerprint of the file. MD5 hashes can be computed with the command md5sum.

- Retrieve original MD5 hash from the metadata

```
../bin/selectRows.sh <(cut -f1 analyses/bigBed.peaks.ids.txt) metadata.tsv | cut -f1,46
> data/bigBed.files/md5sum.txt
```

- Compute MD5 hash on the downloaded files

```
cat data/bigBed.files/md5sum.txt |\
while read filename original_md5sum; do
  md5sum data/bigBed.files/"$filename".bigBed |\
  awk -v filename="$filename" -v original_md5sum="$original_md5sum" 'BEGIN{FS="
"; OFS="\t"}{print filename, original_md5sum, $1}'
done > tmp
mv tmp data/bigBed.files/md5sum.txt
```

- Make sure there are no files for which original and computed MD5 hashes differ

```
awk '$2!=$3' data/bigBed.files/md5sum.txt
```

There are no files for which original and computed MD5 hashes differ.

For each tissue, run an intersection analysis using BEDTools: report:

1) the number of peaks that intersect promoter regions:

- Copy the annotation files of promoters and gene body from the chip-seq to the atac-seq.

cp ../ChIP-seq/annotation/gencode.v24.protein.coding.non.redundant.TSS.bed ATAC-seq/annotation/

cp ../ChIP-seq/annotation/gencode.v24.protein.coding.gene.body.bed ATAC-seq/annotation/

- Make the intersection

```
cut -f-2 analyses/bigBed.peaks.ids.txt |while read filename tissue; do    bedtools intersect -b annotation/gencode.v24.protein.coding.non.redundant.TSS.bed -a data/bed.files/"$filename".bed -u |  wc -l; done
```

47871: sigmoid colon
44749: stomach

2) the number of peaks that fall outside gene coordinates (whole gene body, not just the promoter regions).

```
cut -f-2 analyses/bigBed.peaks.ids.txt |while read filename tissue; do    bedtools intersect -b annotation/gencode.v24.protein.coding.gene.body.bed -a data/bed.files/"$filename".bed -v > peaksoutsidegene"$tissue".bed; done
```

```
wc -l *.bed
```

37035: sigmoid colon
34537: stomach

5. Distal regulatory activity

Task 1: Create a folder regulatory_elements inside epigenomics_uvic. This will be the folder where you store all your subsequent results.

```
mkdir regulatory_elements
cd regulatory_elements
```

Task 2: Distal regulatory regions are usually found to be flanked by both H3K27ac and H3K4me1. From your starting catalogue of open regions in each tissue, select those that overlap peaks of H3K27ac AND H3K4me1 in the corresponding tissue. You will get a list of candidate distal regulatory elements for each tissue. How many are they?

- Create folder in regulatory elements with H3K27acpeaks and H3K4me1peaks

```
mkdir H3K27acpeaks
mkdir H3K4me1peaks
```

- Create bed files with H3K27ac and H3K4me1 for each tissue using the Chip-seq data:

```
grep -F H3K27ac ../ChIP-seq/metadata.tsv |  grep -F "bigBed_narrowPeak" | grep -F
"pseudoreplicated_peaks" |grep -F "GRCh38" |awk 'BEGIN{FS=OFS="\t"}{print $1, $11,
$23}' |sort -k2,2 -k1,1r |sort -k2,2 -u > H3K27acpeaks/bigBed.peaksH3K27ac.ids.txt

cut -f1 H3K27acpeaks/bigBed.peaksH3K27ac.ids.txt |\
while read filename; do
  wget -P H3K27acpeaks
"https://www.encodeproject.org/files/$filename/@@download/$filename.bed"
done


grep -F H3K4me1 ../ChIP-seq/metadata.tsv |  grep -F "bigBed_narrowPeak" | grep -F
"pseudoreplicated_peaks" |grep -F "GRCh38" |awk 'BEGIN{FS=OFS="\t"}{print $1, $11,
$23}' |sort -k2,2 -k1,1r |sort -k2,2 -u > H3K4me1peaks/bigBed.peaksH3K4me1.ids.txt

cut -f1 H3K4me1peaks/bigBed.peaksH3K4me1.ids.txt |\
while read filename; do
  wget -P H3K4me1peaks
"https://www.encodeproject.org/files/$filename/@@download/$filename.bigBed"
done
```

Now we have the bigBed files of both modifications for each tissue. To make the intersect, we have to convert them into bed files.

```
cut -f1 H3K27acpeaks/bigBed.peaksH3K27ac.ids.txt |\
while read filename; do
  bigBedToBed H3K27acpeaks/"$filename".bigBed H3K27acpeaks/"$filename".bed
done
```

```
cut -f1 H3K4me1peaks /bigBed.peaksH3K4me1.ids.txt |\
while read filename; do
  bigBedToBed H3K4me1peaks/"$filename".bigBed H3K4me1peaks/"$filename".bed
done
```

- We do bedtools intersect of H3K4me1peaks with the peaks that are ouside gene coordinates of each tissue:

```
bedtools intersect -a ../ATAC-Seq/peaksoutsidegenesigmoid_colon.bed -b
H3K4me1peaks/ENCFF724ZOF.bed -u > H3K4me1peaks/commonsigmoid.bed
```

```
bedtools intersect -a ../ATAC-Seq/peaksoutsidegenestomach.bed -b
H3K4me1peaks/ENCFF844XRN.bed -u > H3K4me1peaks/commonstomach.bed
```

- Now we do bedtools intersect of the previous intersection with H3K27acpeaks of each tissue.

```
bedtools intersect -a H3K4me1peaks/commonsigmoid.bed -b
H3K27acpeaks/ENCFF872UHN.bed -u > commonsigmoid_colon.bed
```

```
bedtools intersect -a H3K4me1peaks/commonstomach.bed -b
H3K27acpeaks/ENCFF977LBD.bed -u > commonstomach.bed
```

```
wc -l *.bed
```

We see there are 14215 candidate distal regulatory elements for sigmoid colon and 8022 for stomach.

Task 3: Focus on regulatory elements that are located on chromosome 1 (hint: to parse a file based on the value of a specific column, have a look at what we did here), and generate a file regulatory.elements.starts.tsv that contains the name of the regulatory region (i.e. the name of the original ATAC-seq peak) and the start (5') coordinate of the region.

```
cut -f2 ../ATAC-seq/analyses/bigBed.peaks.ids.txt |
while read tissue; do
grep -w chr1 common$tissue.bed | awk 'BEGIN{FS=OFS="\t"}$1=="chr1"{print $4, $2}'
> $tissue.regulatory.elements.starts.tsv
done
```

Task 4: Focus on protein-coding genes located on chromosome 1. From the BED file of gene body coordinates that you generated here, prepare a tab-separated file called gene.starts.tsv which will store the name of the gene in the first column, and the start coordinate of the gene on the second column (REMEMBER: for genes located on the minus strand, the start coordinate will be at the 3'). Use the command below as a starting point:

- Using the file from the ATAC analysis, we will focus on protein-coding genes located on chromosome 1.

```
grep -w chr1 ../ATAC-seq/annotation/gencode.v24.protein.coding.gene.body.bed | awk 'BEGIN{FS=OFS="\t"}{if ($6=="+"){start=$2} else {start=$3}; print $4, start}' >gene.starts.tsv
```

Task 5: Download or copy this python script inside the epigenomics_uvic/bin folder. Have a look at the help page of this script to understand how it works:

```
cat > ../bin/get.distance.py
```

```
#!/usr/bin/env python


#************
# LIBRARIES *
#************

import sys
from optparse import OptionParser


#*****************
# OPTION PARSING *
#*****************

parser = OptionParser()
parser.add_option("-i", "--input", dest="input")
parser.add_option("-s", "--start", dest="start")
options, args = parser.parse_args()

open_input = open(options.input)
enhancer_start = int(options.start)


#********
# BEGIN *
#********

x=1000000 # set maximum distance to 1 Mb
selectedGene="" # initialize the gene as empty
selectedGeneStart=0 # initialize the start coordinate of the gene as empty

for line in open_input.readlines(): # for each line in the input file
                gene, y = line.strip().split('\t') # split the line into two columns based on a tab
                # define a variable called position that correspond to the integer of the start of the gene
                # compute the absolute value of the difference between position and enhancer_start

                # if this absolute value is lower than x
                                # this value will now be your current x
                                # save gene as selectedGene
                                # save position as selectedGeneStart

print "\t".join([selectedGene, str(selectedGeneStart), str(x)])
```

This script takes as input two distinct arguments: 1) --input corresponds to the file gene.starts.tsv (i.e. the file you generated in Task #4); 2) --start corresponds to the 5' coordinate of a regulatory element. Complete the python script so that for a given

coordinate --start the script returns the closest gene, the start of the gene and the distance of the regulatory element.

nano ../bin/get.distance.py

```
#!/usr/bin/env python


#************
# LIBRARIES *
#************

import sys
from optparse import OptionParser


#*****************
# OPTION PARSING *
#*****************

parser = OptionParser()
parser.add_option("-i", "--input", dest="input")
parser.add_option("-s", "--start", dest="start")
options, args = parser.parse_args()
open_input = open(options.input)
enhancer_start = int(options.start)


#********
# BEGIN *
#********

x=1000000 # set maximum distance to 1 Mb
selectedGene="" # initialize the gene as empty
selectedGeneStart=0 # initialize the start coordinate of the gene as empty

for line in open_input.readlines(): # for each line in the input file
      gene, y = line.strip().split('\t') # split the line into two columns based on a tab
      position=int(y) # define a variable called position that correspond to the integer of the start of the gene
      value=position-enhancer_start # compute the absolute value of the difference between position and enhancer_start
      value=abs(value)
      if value<x: # if this absolute value is lower than x
            x=value #this value will now be your current x
                                    selectedGene=gene #save gene as selectedGene
            selectedGeneStart=position #save position as selectedGeneStart

print "\t".join([selectedGene, str(selectedGeneStart), str(x)])
```

Task 6. For each regulatory element contained in the file regulatory.elements.starts.tsv, retrieve the closest gene and the distance to the closest gene using the python script you created above. Use the command below as a starting point:

- **Select the tissue to use as the filename and compute the file to obtain the starts.**

```
cut -f2 ../ATAC-seq/analyses/bigBed.peaks.ids.txt |
while read tissue; do
cat $tissue.regulatory.elements.starts.tsv| while read element start; do
  python ../bin/get.distance.py --input gene.starts.tsv --start $start
done > $tissue.regulatoryElements.genes.distances.tsv
done
```

Task 7: Use R to compute the mean and the median of the distances stored in regulatoryElements.genes.distances.tsv.

R

```
sigmoid<-read.csv("sigmoid_colon.regulatoryElements.genes.distances.tsv", header=F, sep="")
distancesigmoid<-as.vector(unlist(sigmoid[3]))

mean(distancesigmoid, na.rm=TRUE)
73026.44
median(distancesigmoid, na.rm=TRUE)
35768

stomach<-read.csv("stomach.regulatoryElements.genes.distances.tsv", header=F, sep="")
distancestomach<-as.vector(unlist(stomach[3]))

mean(distancestomach)
45227.05
median(distancestomach)
27735
```