5. Distal regulatory activity

Inside `epigenomics_uvic`, run

```
sudo docker run -v $PWD:$PWD -w $PWD --rm -it dgarrimar/epigenomics_course
```

▼ Tasks:

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

```
mkdir regulatory_elements
mkdir regulatory_elements/data
mkdir regulatory_elements/data/tsv
mkdir regulatory_elements/data/bigBed.files
mkdir regulatory_elements/data/bed.files
mkdir regulatory_elements/analyses
mkdir regulatory_elements/analyses/peaks.analyses
```

▼ Select the open regions that overlap peaks of H3K27ac and H3K4me1 in the corresponding tissue. You will get a list of candidate distal regulatory elements for each tissue:

1. Download peak calling files for H3K27ac and H3K4me1:

▼ For H3K27ac:

▼ Get the ids of the bigBed files:

```
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 > regulatory_elements/data/bigBed.files/bigBed.H3K27ac.ids.txt
```

▼ Download the bigBed files:

```
cut -f1 regulatory_elements/data/bigBed.files/bigBed.H3K27ac.ids.txt |\
 while read filename; do
 wget -P regulatory_elements/data/bigBed.files "https://www.encodeproject.org/files/$filename/@@download/$fi
 lename.bigBed"

 done
```

▼ Convert bigBed to bed files:

```
cut -f1 regulatory_elements/data/bigBed.files/bigBed.H3K27ac.ids.txt |\
 while read filename; do
 bigBedToBed regulatory_elements/data/bigBed.files/"$filename".bigBed regulatory_elements/data/bed.files/"$f
 ilename".bed

 done
```

▼ Making sure that my md5sum values coincide with one provided by ENCODE

```
# Define file type
 file_type="bigBed"

 # Loop over each file of the defined type
 for file_type in bigBed; do
     # Define the input file path
     input_file="regulatory_elements/data/bigBed.files/${file_type}.H3K27ac.ids.txt"
     md5sum_file="regulatory_elements/data/${file_type}.files/md5sum_H3K27ac.txt"

     # Check if the input file exists
     if [ -f "$input_file" ]; then
         echo "Processing $input_file..."

         # Run the selectRows.sh script and store the result in md5sum.txt
         if ../bin/selectRows.sh <(cut -f1 "$input_file") ../ChIP-seq/metadata.tsv | cut -f1,46 > "$md5sum_f
 ile"; then
             echo "MD5 checksums are stored in $md5sum_file."
         else
             echo "An error occurred while running selectRows.sh script."
             continue
```

```
        fi
    else
        echo "Input file $input_file does not exist. Skipping this file."
        continue
    fi

    # Check if the md5sum file exists
    if [ -f "$md5sum_file" ]; then
        echo "Starting the verification of $md5sum_file..."

        # Read each line from the md5sum file
        while read -r filename original_md5sum; do
            # Check if the file exists
            if [ -f "regulatory_elements/data/${file_type}.files/${filename}.${file_type}" ]; then
                # Calculate the MD5 checksum of the file and compare it with the original
                calculated_md5sum=$(md5sum "regulatory_elements/data/${file_type}.files/${filename}.${file_
type}" | cut -d ' ' -f 1)

                # Print the filename, original MD5 checksum, and calculated MD5 checksum
                echo -e "$filename\t$original_md5sum\t$calculated_md5sum"
            else
                echo "File regulatory_elements/data/${file_type}.files/${filename}.${file_type} does not ex
ist. Skipping this file."
            fi
        done < "$md5sum_file" > tmp && mv tmp "$md5sum_file"

        echo "The verification of $md5sum_file is completed. The results are stored in $md5sum_file."
    else
        echo "The MD5 checksum file $md5sum_file does not exist. Skipping this file."
    fi
done
```

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

```
awk '$2!=$3' regulatory_elements/data/"$file_type".files/md5sum_H3K27ac.txt

done
```

▼ For H3K4me1:

- ▼ Get the ids of the bigBed files:

```
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 > regulatory_elements/data/bigBed.files/bigBed.H3K4me1.ids.txt
```

- ▼ Download the bigBed files

```
cut -f1 regulatory_elements/data/bigBed.files/bigBed.H3K4me1.ids.txt |\
while read filename; do
wget -P regulatory_elements/data/bigBed.files "https://www.encodeproject.org/files/$filename/@@download/$fi
lename.bigBed"

done
```

- ▼ Convert bigBed to bed files

```
cut -f1 regulatory_elements/data/bigBed.files/bigBed.H3K4me1.ids.txt |\
while read filename; do
bigBedToBed regulatory_elements/data/bigBed.files/"$filename".bigBed regulatory_elements/data/bed.files/"$f
ilename".bed

done
```

- ▼ Making sure that my md5sum values coincide with one provided by ENCODE

```
# Define file type
file_type="bigBed"

# Loop over each file of the defined type
for file_type in bigBed; do
    # Define the input file path
    input_file="regulatory_elements/data/bigBed.files/${file_type}.H3K4me1.ids.txt"
    md5sum_file="regulatory_elements/data/${file_type}.files/md5sum_H3K4me1.txt"

    # Check if the input file exists
    if [ -f "$input_file" ]; then
        echo "Processing $input_file..."

        # Run the selectRows.sh script and store the result in md5sum.txt
        if ../bin/selectRows.sh <(cut -f1 "$input_file") ../ChIP-seq/metadata.tsv | cut -f1,46 > "$md5sum_f
ile"; then
            echo "MD5 checksums are stored in $md5sum_file."
        else
            echo "An error occurred while running selectRows.sh script."
            continue
        fi
    else
        echo "Input file $input_file does not exist. Skipping this file."
        continue
    fi

    # Check if the md5sum file exists
    if [ -f "$md5sum_file" ]; then
        echo "Starting the verification of $md5sum_file..."

        # Read each line from the md5sum file
        while read -r filename original_md5sum; do
            # Check if the file exists
            if [ -f "regulatory_elements/data/${file_type}.files/${filename}.${file_type}" ]; then
                # Calculate the MD5 checksum of the file and compare it with the original
                calculated_md5sum=$(md5sum "regulatory_elements/data/${file_type}.files/${filename}.${file_
type}" | cut -d ' ' -f 1)

                # Print the filename, original MD5 checksum, and calculated MD5 checksum
                echo -e "$filename\t$original_md5sum\t$calculated_md5sum"
            else
                echo "File regulatory_elements/data/${file_type}.files/${filename}.${file_type} does not ex
ist. Skipping this file."
            fi
        done < "$md5sum_file" > tmp && mv tmp "$md5sum_file"

        echo "The verification of $md5sum_file is completed. The results are stored in $md5sum_file."
    else
        echo "The MD5 checksum file $md5sum_file does not exist. Skipping this file."
    fi

done
```

▼ make sure there are no files for which original and computed MD5 hashes differ

```
awk '$2!=$3' regulatory_elements/data/"$file_type".files/md5sum_H3K4me1.txt

done
```

2. Select open regions overlapping with H3K27ac and H3K4me1 in each tissue:

▼ The Bash script intersects ATAC-seq peaks with H3K27ac and H3K4me1 marks for each tissue, then finds the common peaks between H3K27ac and H3K4me1, and finally counts the number of peaks in each tissue-specific file and in all files combined.

💡 The `peak_analysis.sh` script is located in the `bin` directory under the `ATAC-seq` project, which is a part of the `epigenomics_uvic` study.

```
#!/bin/bash

# Define the list of tissues
tissues=("sigmoid_colon" "stomach")

# Print a header
echo "==================== Peak Analysis ===================="

# Loop over each tissue
for tissue in "${tissues[@]}"; do
    echo "-----------------------------------------------------"
    echo "Processing $tissue..."

    # Intersect ATAC-seq peaks with H3K27ac mark
    echo "Intersecting ATAC-seq peaks with H3K27ac mark for $tissue..."
    bedtools intersect -a analyses/peaks.analysis/peaks.not.body."$tissue".bed -b regulatory_elements/data/be
d.files/$(awk -v tissue="$tissue" '$2 == tissue {print $1}' regulatory_elements/data/bigBed.files/bigBed.H3K27
ac.ids.txt).bed -u > regulatory_elements/analyses/peaks.analyses/peaks.regulatory.H3K27ac."$tissue".bed

    echo "Done with H3K27ac intersection for $tissue."

    # Intersect ATAC-seq peaks with H3K4me1 mark
    echo "Intersecting ATAC-seq peaks with H3K4me1 mark for $tissue..."
    bedtools intersect -a analyses/peaks.analysis/peaks.not.body."$tissue".bed -b regulatory_elements/data/be
d.files/$(awk -v tissue="$tissue" '$2 == tissue {print $1}' regulatory_elements/data/bigBed.files/bigBed.H3K4m
e1.ids.txt).bed -u > regulatory_elements/analyses/peaks.analyses/peaks.regulatory.H3K4me1."$tissue".bed

    echo "Done with H3K4me1 intersection for $tissue."

    # Intersect H3K27ac and H3K4me1 peaks
    echo "Intersecting H3K27ac and H3K4me1 peaks for $tissue..."
    bedtools intersect -a regulatory_elements/analyses/peaks.analyses/peaks.regulatory.H3K4me1."$tissue".bed -
b regulatory_elements/analyses/peaks.analyses/peaks.regulatory.H3K27ac."$tissue".bed -u > regulatory_elements/
analyses/peaks.analyses/peaks.regulatory.H3K4me1.AND.H3K27ac."$tissue".bed

    echo "Done with H3K27ac and H3K4me1 intersection for $tissue."

    # Count the number of lines (peaks) in each file
    num_peaks=$(wc -l < regulatory_elements/analyses/peaks.analyses/peaks.regulatory.H3K4me1.AND.H3K27ac."$tis
sue".bed)
    echo "The file for $tissue contains $num_peaks peaks."
done

# Count the total number of lines (peaks) in all files
total_peaks=$(cat regulatory_elements/analyses/peaks.analyses/peaks.regulatory.H3K4me1.AND.H3K27ac.*.bed | wc
-l)
echo "-----------------------------------------------------"
echo "The total number of peaks in all files is $total_peaks."

echo "==================== All done! ===================="
```

▼ Results:

```
==================== Peak Analysis ====================
-----------------------------------------------------
Processing sigmoid_colon...
Intersecting ATAC-seq peaks with H3K27ac mark for sigmoid_colon...
Done with H3K27ac intersection for sigmoid_colon.
Intersecting ATAC-seq peaks with H3K4me1 mark for sigmoid_colon...
Done with H3K4me1 intersection for sigmoid_colon.
Intersecting H3K27ac and H3K4me1 peaks for sigmoid_colon...
Done with H3K27ac and H3K4me1 intersection for sigmoid_colon.

The file for sigmoid_colon contains 14215 peaks.
-----------------------------------------------------
Processing stomach...
Intersecting ATAC-seq peaks with H3K27ac mark for stomach...
Done with H3K27ac intersection for stomach.
Intersecting ATAC-seq peaks with H3K4me1 mark for stomach...
Done with H3K4me1 intersection for stomach.
Intersecting H3K27ac and H3K4me1 peaks for stomach...
Done with H3K27ac and H3K4me1 intersection for stomach.

The file for stomach contains 8022 peaks.
-----------------------------------------------------
The total number of peaks in all files is 22237.
==================== All done! ====================
```

▼ Focus on the regulatory elements that are located on chromosome 1 and generate a file `regulatory.elements.starts.tsv` that contains the name of the regulatory region and the start (5') coordinate of the region:

```
# Define the list of tissues
tissues=("sigmoid_colon" "stomach")

# Loop over each tissue
for tissue in "${tissues[@]}"; do
    # Print a message to let the user know which tissue is being processed
    echo "Processing $tissue..."

    # Use awk to filter for lines where the first field is "chr1", then print the fourth and second fields
    # Redirect the output to the appropriate .tsv file
    awk 'BEGIN{FS=OFS="\t"} $1=="chr1" {print $4, $2}' regulatory_elements/analyses/peaks.analyses/peaks.regulatory.
```

```
H3K4me1.AND.H3K27ac."$tissue".bed > regulatory_elements/data/tsv/regulatory.elements.starts."$tissue".tsv
done

echo "All done!"
```

▼ Focus on protein-coding genes located on chromosome 1. From the BED file of the gene body coordinated that you generated, prepare a tab-separated file called gene.starts.tsv which will store the name of the gene in the first column, the start coordinate of the gene on the second column:

```
# Extract gene starts for protein-coding genes on chromosome 1
awk 'BEGIN{FS=OFS="\t"} $1=="chr1" {start = ($6=="+") ? $2 : $3; print $4, start}' ../ChIP-seq/annotation/gencode.v2
4.protein.coding.gene.body.bed > regulatory_elements/data/tsv/gene.starts.tsv

done
```

▼ Download or copy the python script inside `epigenomics_uvic/bin` folder. Have a look at the help page of the script to undertand how it works:

```
#!/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
    diff = abs(position - enhancer_start) # compute the absolute value of the difference between position and enhanc
er_start

    if diff < x: # if this absolute value is lower than x
        x = diff # 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)])
```

💡 To make sure your script is working fine, run the following command:

```
python ../bin/get.distance.py --input regulatory_elements/data/tsv/gene.starts.tsv --start 98000
```

▼ Finding the closest gene and the distance to the closest gene for each regulatory element contained in the file `regulatory.elements.tsv.`

💡 Using the python script `get.distance.py`, it retrieves the closest gene and the distance to the closest gene for each regulatory element and for each tissue

```
# Define the list of tissues
tissues=("sigmoid_colon" "stomach")
```

```
# Loop over each tissue
for tissue in "${tissues[@]}"; do
    echo "Processing $tissue..."

    # Find the closest gene and the distance to the closest gene for each regulatory element
    while read element start; do
        python ../bin/get.distance.py --input regulatory_elements/data/tsv/gene.starts.tsv --start $start
    done < regulatory_elements/data/tsv/regulatory.elements.starts."$tissue".tsv > regulatory_elements/data/tsv/regu
latoryElements.genes.distances."$tissue".tsv

done
```

▼ Computing the mean and median of the distances to the closest gene for each tissue:

```
# Load the data
regulatory.sigmoid_colon <- read.table(file = '/epigenomics_uvic/ATAC-seq/regulatory_elements/data/tsv/regulatoryEle
ments.genes.distances.sigmoid_colon.tsv', sep = '\t', header = FALSE)

regulatory.stomach <- read.table(file = '/epigenomics_uvic/ATAC-seq/regulatory_elements/data/tsv/regulatoryElements.
genes.distances.stomach.tsv', sep = '\t', header = FALSE)

# Compute the mean and median distances for each tissue
mean.sigmoid_colon <- mean(regulatory.sigmoid_colon$V3)
median.sigmoid_colon <- median(regulatory.sigmoid_colon$V3)

mean.stomach <- mean(regulatory.stomach$V3)
median.stomach <- median(regulatory.stomach$V3)

# Print the results
cat("Mean distance for sigmoid colon: ", mean.sigmoid_colon)
cat("Median distance for sigmoid colon: ", median.sigmoid_colon)

cat("Mean distance for stomach: ", mean.stomach)
cat("Median distance for stomach: ", median.stomach)
```

▼ Results:

```
Print the results

cat("Mean distance for sigmoid colon: ", mean.sigmoid_colon)
Mean distance for sigmoid colon:  73635.89
cat("Median distance for sigmoid colon: ", median.sigmoid_colon)
Median distance for sigmoid colon:  35802

cat("Mean distance for stomach: ", mean.stomach)
Mean distance for stomach:  45227.05
cat("Median distance for stomach: ", median.stomach)
Median distance for stomach:  27735
```

Tissue	**Mean**	**Median**
Sigmoid_colon	73635.89	35802
Stomach	45227.05	27735