ATAC-seq Module2: Visualization and Peak Identification

Drawing

Overview & Purpose

In the previous section of this module we performed preprocessing quality control, mapping, and deduplication. In this section we will focus on visualization of the signal, create average plots of signal around transcription start sites (TSSs), and identification of peak signal.

Required Files

In this stage of the module you will use the deduplicated bam files that we prepared in the previous section. Don't worry if you are just jumping in now, we have examples of these files saved and will include a step that copies them for your use. You can also use this module on your own data or any published ATAC-seq dataset, but you should complete the mappping and deduplication steps first.

STEP1: Setup Environment

Initial items to configure your google cloud environment. In this step we will use conda to install the following packages:

Visualization: samtools, deeptools, IGV

Peak Identification: macs2

In [9]:
```
#!python -m ipykernel install --user --name ATACtraining
numthreads=!lscpu | grep '^CPU(s)'| awk '{print $2-1}'
numthreadsint = int(numthreads[0])
```

In [10]:
```
numthreadsint
```

Out[10]:
```
7
```

In [1]:
```
!conda install -y -c conda-forge ncurses
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.
```

In [2]:
```
!conda config --prepend channels bioconda
```

```
Warning: 'bioconda' already in 'channels' list, moving to the top
```

In [3]:
```
#!python -m pip install --user --upgrade pdf2image
#from pdf2image import convert_from_path, convert_from_bytes
!conda install -y -c bioconda samtools deeptools macs2
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - deeptools
    - macs2
    - samtools


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    deeptools-3.5.1            |             py_0         143 KB  bioconda
    deeptoolsintervals-0.1.9   |  py310h8472f5a_5          76 KB  bioconda
    macs2-2.2.9.1              |  py310h4b81fae_0         1.5 MB  bioconda
    plotly-5.16.1              |     pyhd8ed1ab_0         5.7 MB  conda-forge
    pooch-1.7.0                |     pyha770c72_3          50 KB  conda-forge
    py2bit-0.3.0               |  py310h4b81fae_8          25 KB  bioconda
    pybigwig-0.3.22            |  py310h79000e5_1          89 KB  bioconda
    pysam-0.21.0               |  py310h41dec4a_1         4.1 MB  bioconda
    scipy-1.11.2               |  py310ha4c1d20_0        14.8 MB  conda-forge
    tenacity-8.2.3             |     pyhd8ed1ab_0          22 KB  conda-forge
    ------------------------------------------------------------
                                           Total:        26.6 MB

The following NEW packages will be INSTALLED:

  deeptools          bioconda/noarch::deeptools-3.5.1-py_0
  deeptoolsintervals bioconda/linux-64::deeptoolsintervals-0.1.9-py310h8472f5a_5
  macs2              bioconda/linux-64::macs2-2.2.9.1-py310h4b81fae_0
  plotly             conda-forge/noarch::plotly-5.16.1-pyhd8ed1ab_0
  pooch              conda-forge/noarch::pooch-1.7.0-pyha770c72_3
  py2bit             bioconda/linux-64::py2bit-0.3.0-py310h4b81fae_8
  pybigwig           bioconda/linux-64::pybigwig-0.3.22-py310h79000e5_1
  pysam              bioconda/linux-64::pysam-0.21.0-py310h41dec4a_1
  scipy              conda-forge/linux-64::scipy-1.11.2-py310ha4c1d20_0
  tenacity           conda-forge/noarch::tenacity-8.2.3-pyhd8ed1ab_0



Downloading and Extracting Packages
pysam-0.21.0         | 4.1 MB    |                                       |   0%
py2bit-0.3.0         | 25 KB     |                                       |   0%

macs2-2.2.9.1        | 1.5 MB    |                                       |   0%


plotly-5.16.1        | 5.7 MB    |                                       |   0%



scipy-1.11.2         | 14.8 MB   |                                       |   0%




pybigwig-0.3.22      | 89 KB     |                                       |   0%





deeptoolsintervals-0 | 76 KB     |                                       |   0%






deeptools-3.5.1      | 143 KB    |                                       |   0%







pooch-1.7.0          | 50 KB     |                                       |   0%








tenacity-8.2.3       | 22 KB     |                                       |   0%

pysam-0.21.0         | 4.1 MB    | 1                                     |   0%
macs2-2.2.9.1        | 1.5 MB    | 3                                     |   1%

plotly-5.16.1        | 5.7 MB    | 1                                     |   0%

pysam-0.21.0         | 4.1 MB    | ###########2                          |  31%
macs2-2.2.9.1        | 1.5 MB    | #######################6              |  64%

pysam-0.21.0         | 4.1 MB    | ##############################3       |  82%

scipy-1.11.2         | 14.8 MB   | #########1                            |  25%

pybigwig-0.3.22      | 89 KB     | ######6                               |  18%
py2bit-0.3.0         | 25 KB     | #######################7              |  64%

plotly-5.16.1        | 5.7 MB    | ######################                |  60%

deeptoolsintervals-0 | 76 KB     | #######7                              |  21%

scipy-1.11.2         | 14.8 MB   | ############7                         |  34%

pooch-1.7.0          | 50 KB     | ###########9                          |  32%

plotly-5.16.1        | 5.7 MB    | ##############################6       |  83%

tenacity-8.2.3       | 22 KB     | ##########################5           |  72%
py2bit-0.3.0         | 25 KB     | ##################################### | 100%
py2bit-0.3.0         | 25 KB     | ##################################### | 100%

scipy-1.11.2         | 14.8 MB   | ################1                     |  44%

pybigwig-0.3.22      | 89 KB     | ##################################### | 100%

pybigwig-0.3.22      | 89 KB     | ##################################### | 100%

deeptools-3.5.1      | 143 KB    | ####1                                 |  11%

scipy-1.11.2         | 14.8 MB   | ###################7                  |  53%

deeptoolsintervals-0 | 76 KB     | ##################################### | 100%

deeptoolsintervals-0 | 76 KB     | ##################################### | 100%

scipy-1.11.2         | 14.8 MB   | ###########################           |  73%

scipy-1.11.2         | 14.8 MB   | ################################6     |  88%

pooch-1.7.0          | 50 KB     | ##################################### | 100%

pooch-1.7.0          | 50 KB     | ##################################### | 100%

tenacity-8.2.3       | 22 KB     | ##################################### | 100%

tenacity-8.2.3       | 22 KB     | ##################################### | 100%
macs2-2.2.9.1        | 1.5 MB    | ##################################### | 100%

deeptools-3.5.1      | 143 KB    | ##################################### | 100%

pysam-0.21.0         | 4.1 MB    | ##################################### | 100%

scipy-1.11.2         | 14.8 MB   | ##################################### | 100%

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

In [29]:
```
#!python -m pip install --user --upgrade macs3
#!conda install -y -c maximinio macs3



# !python -m pip install --user --upgrade numpy numpydoc
# !pip install jupyterquiz
# !pip install --user igv-notebook
import sys
# # sys.path.insert(0, "/home/jupyter/.local/lib/python3.7/site-packages")
sys.path.insert(0,  "/home/jupyter/.local/lib/python3.10/site-packages")

# igv-notebook in ./.local/lib/python3.10/site-packages
```

In [20]:
```
import igv_notebook
```

In [4]:
```
from jupyterquiz import display_quiz
from IPython.display import IFrame
from IPython.display import display
from IPython.display import Image
import pandas as pd
```

Setup FileSystem

Now lets create some folders to stay organized and copy over our prepared fastq files. We're going to create a directory called "Tutorial1" which we'll use for this module. We'll then create subfolders for our InputFiles and for the files that we'll be creating during this module. We'll also copy over the fasta file for chromosome 4 as well as some bowtie2 index files (don't worry we'll teach you how to create these index files).

In [11]:
```
#These commands create our directory structure.
#!cd $HOMEDIR
#!mkdir -p Tutorial2
#!mkdir -p Tutorial2/InputFiles
#!mkdir -p Tutorial2/GenomeAnnotations
!mkdir -p Tutorial2/BigWigFiles
!mkdir -p Tutorial2/Peaks
#!mkdir -p Tutorial2/LessonImages
!mkdir -p Tutorial2/Plots
#!cd ./Tutorial2
#!echo $PWD

#These commands help identify the google cloud storage bucket where the example files are held.
#project_id = "nosi-unmc-seq"
#original_bucket = "gs://unmc_atac_data_examples/Tutorial2"
#!gsutil -m cp $original_bucket/images/* Tutorial2/LessonImages
#!gsutil -m cp $original_bucket/Annotations/* Tutorial2/GenomeAnnotations
#This command copies our example files to the Tutorial1/Inputfiles folder that we created above.
#!gsutil -m cp $original_bucket/InputFiles/*bam Tutorial2/InputFiles
```

OK

Let's make sure that the files copied correctly. You should see 2 .bam files after running the following command:

In [12]:
```
!ls Tutorial2/InputFiles
```

```
CTL_dedup.bam  Mutant_dedup.bam
```

STEP2: Visualization

Files in sam/bam format contain a lot of information including the original sequence of the reads, quality scores, and their corresponding chromosomal coordinates.

Drawing

Please view this site for a more complete description of sam format and to see what the various sam flag values mean.

Let's view the first few lines of one of our bam files:

In [13]:
```
!samtools view Tutorial2/InputFiles/CTL_dedup.bam | head -3
#Note that there will be an error message because we are breaking a pipe by printing only the first 3 lines. Pl
```

```
SRR1944627.37127681     99      chr4    39845   31      50M     =       39881   86      ATCTTTGTGGCATTCTCTGTATT
TCCTGAATTTGAATGTTGGCCTGCCTT     CCCFFFFFHHHHHJJJJJJJJJJJJJJJJJJIJIIIJJIJJJJJJJJJJ      MD:Z:50 PG:Z:MarkDuplic
ates    XG:i:0  NM:i:0  XM:i:0  XN:i:0  XO:i:0  AS:i:0  XS:i:0  YS:i:0  YT:Z:CP
SRR1944627.37127681     147     chr4    39881   31      50M     =       39845   -86     TGTTGGCCTGCCTTGCTAGGTTG
GGAAAGTTCTCCTGGATAATATCCTGA     HEJJJIJJJJJJJJHJJJJJJIJJJJJJIJJIJJJJJHHHHHFFFFFCCC      MD:Z:50 PG:Z:MarkDuplic
ates    XG:i:0  NM:i:0  XM:i:0  XN:i:0  XO:i:0  AS:i:0  XS:i:0  YS:i:0  YT:Z:CP
SRR1944627.50776065     99      chr4    98978   11      49M     =       99304   376     GAGTCTCACTCTGTCACCCAGGC
TGGAGTGCAGTGGCACGATCTCGGCT      @C@FFFFFHHGHGHIIJJJJIGIBHEIICFCHGEHJGIJIHCHIIJGII       MD:Z:49 PG:Z:MarkDuplic
ates    XG:i:0  NM:i:0  XM:i:0  XN:i:0  XO:i:0  AS:i:0  XS:i:0  YS:i:-12        YT:Z:CP
samtools view: writing to standard output failed: Broken pipe
samtools view: error closing standard output: -1
```

While we can see the coordinates of each read, we will need a better way of visualizing the results. In this step we will create a binary file that summarizes the pileup of reads at basepair along our genome, in bigwig format.

To create the bigwig files let's use the command bamCoverage, part of the deeptools package.

In [ ]:

In [14]:
```
# First we need to create an index of our bam file.
!samtools index Tutorial2/InputFiles/CTL_dedup.bam
```

In [36]:
```
# Then we can create a bigwig file of the control sample.
!bamCoverage -b Tutorial2/InputFiles/CTL_dedup.bam -o Tutorial2/BigWigFiles/Control.bw -bs 1 -p $numthreadsint
```

In [37]:
```
# Now let's rerun the commands for our mutant sample.
!samtools index Tutorial2/InputFiles/Mutant_dedup.bam
!bamCoverage -b Tutorial2/InputFiles/Mutant_dedup.bam -o Tutorial2/BigWigFiles/Mutant.bw -bs 1 -p $numthreadsin

print("done")
```

```
done
```

In the above example we specify the bam file name after -b and the output file name after -o.

We specified -bs 1, which tells bamCoverage the summarize the reads at every basepair; the default is to summarize at 50 bp resolution, but for ATAC-seq we find it useful to summarize the data at finer-scale.

We also specified the number of threads to use with -p, which is held in a variable in our notebook.

Lastly, we specified --normalizeUsing BPM. BPM stands for Bins Per Million mapped reads. What do you think this normalization does?

Interactive Quiz Question 1: Click on the correct answer in following cell.

In [17]:
```
display_quiz("Tutorial2/LessonImages/BPMnorm.json")
```

Why do you think we BPM normalize the signal?

This will remove PCR duplicates.

This will remove repetitive regions.

We are normalizing to account for the total number of reads that we sequenced.

We are normalizing for differing depths of coverage.

Genome Browser

Now that we have our bigwig files, we can visualize the signal in a genome browser. We'll use igv in this example.

In [31]:
```
# !pip install igv-notebook
```

In [38]:
```
igv_notebook.init()
myigv = igv_notebook.Browser(
    {
        "genome": "hg38",
        "locus": "chr4:55,400,000-56,400,000"
    }
)
myigv.load_track(
{
        "name": "CTL",
        "url": "Tutorial2/BigWigFiles/Control.bw",
        "format": "bigwig",
        "type": "wig"
    }

)
myigv.load_track(
{
        "name": "Mutant",
        "url": "Tutorial2/BigWigFiles/Mutant.bw",
        "format": "bigwig",
        "type": "wig"
    }

)
```

This will load in the signal into IGV and allow you to browse the genome. Feel free to play around with this. More instructions can be found on the IGV website.

Notice that when we first load in the files, the scales are different on the left hand side. IGV defaults to autoscale each individually. However, if we want to compare to signals we should use the same y-axis scale for both. We can do this because we included BPM normalization. To change the scale, click on the gear icon on the right of each track and select "Set data range". Let's set the maximum to 300 both both.

Drawing

In addition to scrolling along the genome, go ahead an try to zoom in on a specific "peak" of signal. You can do so by clicking on the top ruler (where the coordinates are displayed), holding, and dragging either direction. Alternatively, you can click on the + and - signs at the top right.

Average Profiles

In addition to browsing, we can make average profiles of signal across specific regions. For example, ATAC-seq signal should be enriched near TSSs. Let's test this using deeptools.

Deeptools takes in a bigwig file representing the signal. It also takes a bed file representing the features across which one wants to average the signal. In our case the bed file will be composed of gene annotations. Creating the profile will occur in two steps. The first is to create the summarized matrix, while the second plots that data.

In [1]:
```
#-S option specifies the bigwig signal file, where we can specify multiple separated by spaces. -R option spec
!computeMatrix reference-point --referencePoint TSS -S Tutorial2/BigWigFiles/Control.bw Tutorial2/BigWigFiles/
```

In [2]:
```
!plotProfile -m Tutorial2/Plots/TSSprofileMatrix -o Tutorial2/Plots/TSSprofile.png
```

Let's view the output:

In [5]:
```
Image(url= "Tutorial2/Plots/TSSprofile.png", width=400, height=400)
```

Out[5]:

A note on insert sizes

As reported in the original ATAC-seq publication, high quality ATAC-seq datasets reveal a specific distribution of insert sizes that correspond to distinct chromatin features. To view an example of this distribution, see the following publication: Buenrostro et al., Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position, Nat. Methods, 2013.

Drawing

Here we see abundant insert sizes corresponding to accessible chromatin vs nucleosomal fragments. Note that we only know the insert size with Paired-end data, and not with single-end sequencing.

Nucleosomes consist of 145 bp of DNA wrapped around histones. Because Tn5 randomly inserts near protected sites, in paired-end ATAC-seq this results in a slightly larger range of protected fragments (i.e. insertion sizes). Based on this information, look at the graph and think about the size range that would be most consistent with TF binding vs mono-nucleosomes.

We can use Deeptools to summarize our insert sizes.

In [11]:
```
!bamPEFragmentSize -b Tutorial2/InputFiles/CTL_dedup.bam Tutorial2/InputFiles/Mutant_dedup.bam -o Tutorial2/Pl
print("done")
```

```
done
```

In [14]:
```
Image(url= "Tutorial2/Plots/Insertsizes_histogram.png", width=400, height=400)
```

Out[14]:

Interactive Quiz Question 2: Click on the correct answer in following cell.

In [13]:
```
display_quiz("Tutorial2/LessonImages/InsertSizeQuiz.json")
```

Based on our knowledge of the ATAC-seq insert distributions, which of the following would be an appropriate insert size range to obtain mononucleosomal fragments?

<150

150-175

150-250

300-500

With paired-end ATAC-seq data we can separate by fragment size to obtain Transposase HyperSensitive Sites (THSS) and Nucleosomal Fragments. Alternatively, some choose to keep the data together as a more general measure of "accessible" sites.

We'll show you how to separate the small and large fragments into different bam files.

In [15]:
```
#Filter by insert size:
!samtools view -h Tutorial2/InputFiles/CTL_dedup.bam | awk 'substr($0,1,1)=="@" || ($9>= 150 && $9<=250) || (
!samtools view -h Tutorial2/InputFiles/CTL_dedup.bam | awk 'substr($0,1,1)=="@" || ($9>= 10 && $9<=125) || ($
#Do the same for the mutant:
!samtools view -h Tutorial2/InputFiles/Mutant_dedup.bam | awk 'substr($0,1,1)=="@" || ($9>= 150 && $9<=250) |
!samtools view -h Tutorial2/InputFiles/Mutant_dedup.bam | awk 'substr($0,1,1)=="@" || ($9>= 10 && $9<=125) ||
```

For the rest of this tutorial, we'll use the bam files that contain all the reads as many use this as a general measurement of "accessibility". However, you can use these split bam files to create bigwigs, view them in a genome browser, and create average profiles around features as demonstrated earlier. You can also use them in our downstream analysis in lieu of the combined file that we will show in our examples.

STEP3: Peak Detection

Drawing

Accessible sites are loci with a pileup of reads in "Peaks".

Opitional Note:

Tn5 insertion of adapters leaves a 9 bp gap. In the end, this probably won't impact the results much. However, to be safe we can shift the reads to account for this insertion offset.

Drawing

Image adjusted from: Grandi et al., Nature Protocols 2022

The alignmentSieve command from deeptools allows us to shift the reads accordingly.

In [16]:
```
!alignmentSieve -p $numthreadsint --ATACshift -b Tutorial2/InputFiles/CTL_dedup.bam -o Tutorial2/InputFiles/C
!alignmentSieve -p $numthreadsint --ATACshift -b Tutorial2/InputFiles/Mutant_dedup.bam -o Tutorial2/InputFile
```

Let's identify Peaks genome-wide using macs2.

In [17]:
```
#If your data is single-end (not paired-end), use -f BAM instead.
!macs2 callpeak -f BAMPE -g hs --keep-dup all --cutoff-analysis -n CTL -t Tutorial2/InputFiles/CTL_shift.bam
!macs2 callpeak -f BAMPE -g hs --keep-dup all --cutoff-analysis -n Mutant -t Tutorial2/InputFiles/Mutant_shif
```

macs2 provides a .narrowPeak file specififying the coordinates of the peaks, an .xls file with additional information, and a .bed file with the summits of the peaks.

Let's view the first 10 lines of the .narrowPeak file.

In [18]:
```
!head Tutorial2/Peaks/CTL_peaks.narrowPeak
```

```
chr4    4098436 4098780 CTL_peak_1      23      .       2.90799 5.28767 2.33528 172
chr4    26975641        26975876        CTL_peak_2      23      .       2.90799 5.28767 2.33528 117
chr4    49751053        49751289        CTL_peak_3      23      .       2.90799 5.28767 2.33528 118
chr4    49771937        49772236        CTL_peak_4      157     .       11.172  19.3961 15.7605 148
chr4    49803060        49803221        CTL_peak_5      20      .       2.88586 4.99951 2.07827 80
chr4    49842974        49843212        CTL_peak_6      23      .       2.90799 5.28767 2.33528 119
chr4    49927479        49927778        CTL_peak_7      36      .       3.81875 6.69129 3.6285  200
chr4    50048359        50048660        CTL_peak_8      44      .       4.63347 7.61679 4.49353 209
chr4    50589614        50589840        CTL_peak_9      23      .       2.90799 5.28767 2.33528 113
chr4    50622209        50622416        CTL_peak_10     18      .       2.86406 4.76454 1.86719 103
```

We can also visually inspect the peaks compared to the signal in igv:

In [21]:
```
igv_notebook.init()
```

In [22]:
```
igv_notebook.init()
myigv = igv_notebook.Browser(
    {
        "genome": "hg38",
        "locus": "chr4:55,570,000-55,670,000"
    }
)
myigv.load_track(
{
        "name": "CTL",
        "url": "Tutorial2/BigWigFiles/Control.bw",
        "format": "bigwig",
        "type": "wig"
    }

)
myigv.load_track(
{
        "name": "CTL_peaks",
        "url": "Tutorial2/Peaks/CTL_peaks.narrowPeak",
        "format": "bed",
        "type": "annotation"
    }

)
myigv.load_track(
{
        "name": "Mutant",
        "url": "Tutorial2/BigWigFiles/Mutant.bw",
        "format": "bigwig",
        "type": "wig"
    }

)
myigv.load_track(
{
        "name": "Mutant_peaks",
        "url": "Tutorial2/Peaks/Mutant_peaks.narrowPeak",
        "format": "bed",
        "type": "annotation"
    }

)
```

Great job!

We have completed the first downstream processing steps and are ready to move on to some additional downstream analysis. Take a break here or move on to the next tutorial.

Downtream Analysis

In [ ]: