ChIP–Seq Peak Calling

Saurabh Sinha

PowerPoint by Shayan Tabe Bordbar

Introduction

This goals of the lab are as follows:

1. Learn how to map Next Generation Sequencing (NGS) reads to a reference genome using **Bowtie2.**

2. Demonstrate how to call peaks from aligned reads (in SAM format) using **MACS2**.

Start the VM

- Follow instructions for starting VM. (This is the Remote Desktop software.)

- The instructions are different for UIUC and Mayo participants.

- Instructions for UIUC users are here: http://publish.illinois.edu/compgenomicscourse/files/2020/06/SetupVM_UIUC.pdf

- Instructions for Mayo users are here:

http://publish.illinois.edu/compgenomicscourse/files/2020/06/VM_Setup_Mayo.pdf

Step 0A: Accessing the IGB Biocluster

Open **Putty.exe**

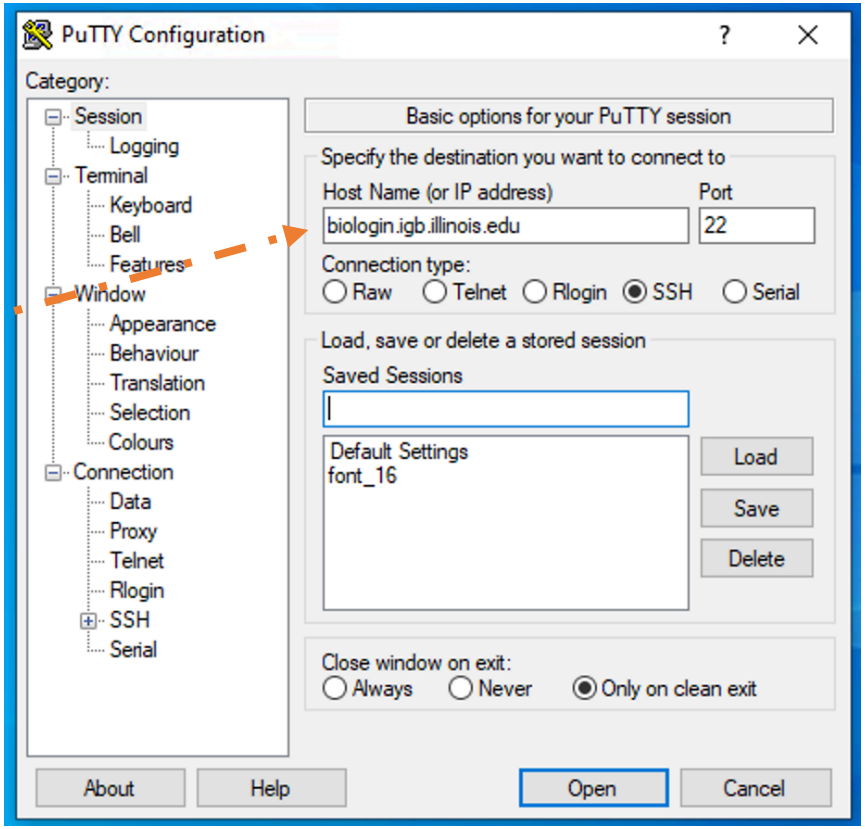
In the **hostname** textbox type:

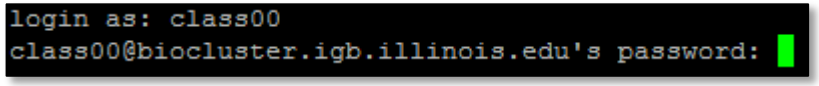
`biologin.igb.illinois.edu`

Click **Open**

If popup appears, Click **Yes**

Enter login credentials assigned to you; example, user **class00**.





Now you are all set!

Step 0B: Lab Setup

The lab is located in the following directory:

`/home/classroom/mayo/2020/05_Epigenomics/`

Following commands will copy a shell script -designed to prepare the working directory- to your home directory. Follow these steps to copy and then submit the script as a job to biocluster:

```
$ cd ~/

# Note ~ is a symbol in Unix paths referring to your home directory

$ cp /home/classroom/mayo/2020/05_Epigenomics/src/prep-directory.sh ./

# Copies prep-directory.sh script to your working directory.

$ sbatch prep-directory.sh

# submits a job to biocluster to populate your home directory with necessary
files

$ squeue –u <userID> # to check the status of the submitted job
```

Step 0C: Working directory: data

Navigate to the created directory for this exercise and look what data folder contains.

```
$ cd 05_Epigenomics

$ ls

# output should be:

# data results src

$ ls data/

$ ls data/index
```

Note: G1E cell lines are erythroid, red blood cell, cell lines missing the GATA-1 gene.

GATA-1 is crucial for the maturation of erythroid cells.

G1E_E4R cell lines conditionally express GATA-1 in the presence of estradiol, enabling erythroid maturation.

Filename	Description
G1E_ER4_CTCF_chr19.fastqsanger	A sample ChIP-seq dataset on CTCF in G1E_ER4 cells, reads have been reduced to those mapping to chr19 for demonstration use.
G1E_ER4_input_chr19.fastqsanger	Control DNA taken from chr19.
G1E_CTCF.fastqsanger	CTCF Chip for G1E line.
G1E_input.fastqsanger	Control for G1E line.

Step 0D: Working directory: scripts

Navigate to the directory containing the scripts and look what's inside.

```
$ cd src
$ ls *.sh
# lists the scripts to be used in this lab:
#      fastx_summary.sh
#      run_bowtie2.sh
#      run_macs2_noControl.sh      run_macs2_withControl.sh    run_macs2_noER.sh
#      bedtools_overlap_1.sh  bedtools_overlap_2.sh
#      bedtools_subtract_1.sh  bedtools_subtract_2.sh
```

Read Mapping and Peak Calling

In this exercise, we will map ChIP Reads to a reference genome using **Bowtie2** and call peaks among the mapped reads using **MACS2**.

Step 1: FASTQ Summary Statistics

In this step, we will gather summary statistics of ChIP data for quality control.

We use FASTX-Toolkit to get statistics on the quality and content of each column of fastq files (sequencing reads).

“fastx_quality_stats” is the name of the tool used from FASTX-Toolkit.

How to Use [Do not run the following command]:

```
$ fastx_quality_stats    -i <input.fastq> -o <output_summary.txt>
```

fastx_summary.sh uses fastx_quality_stats to get summary reports for all four provided fastq files. Run the following command:

```
$ cd ~/05_Epigenomics/src/

$ sbatch fastx_summary.sh

# OUTPUT in ~/05_Epigenomics/results/

$ squeue –u <userID> # to check the status of the submitted job
```

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (fastx_summary.sh) is supposed to do in more detail.

What's inside the **fastx_summary.sh** script?

```
#!/bin/bash
#SBATCH -c 4
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J fastx_summ
#SBATCH -o fastx_summ.%j.out
#SBATCH -e fastx_summ.%j.err
#SBATCH -p classroom


# load the tool environment
module load FASTX-Toolkit

# this is our input (fastq)
export FASTQ_1=../data/G1E_CTCF.fastqsanger
export FASTQ_2=../data/G1E_input.fastqsanger
export FASTQ_3=../data/G1E_ER4_CTCF_chr19.fastqsanger
export FASTQ_4=../data/G1E_ER4_input_chr19.fastqsanger
# this is our output (summaries)
export OUT_1=../results/G1E_CTCF_summary.txt
export OUT_2=../results/G1E_input_summary.txt
export OUT_3=../results/G1E_ER4_CTCF_chr19_summary.txt
export OUT_4=../results/G1E_ER4_input_chr19_summary.txt

fastx_quality_stats -i $FASTQ_1  -o $OUT_1
fastx_quality_stats -i $FASTQ_2  -o $OUT_2
fastx_quality_stats -i $FASTQ_3  -o $OUT_3
fastx_quality_stats -i $FASTQ_4  -o $OUT_4
```

Tells the cluster 'job manager' what resources you want (4 CPUs, 8GB memory, run on the 'classroom' nodes, and name the job 'fastx_summ'

Load the software. We are using a tool called 'FASTX-Toolkit' to generate some basic stats on the fastq files.

Create shortcut names for input and output files.

Run the tool on all four input fastq files.

Step 1: FASTQ Summary Statistics

- To view one of the summary files use:

```
$ more ~/05_Epigenomics/results/G1E_CTCF_summary.txt

# shows beginning of G1E_CTCF_summary.txt file

# To get the length of the reads use:

$ cat ~/05_Epigenomics/results/G1E_CTCF_summary.txt | wc -l

# you should get 37
```

Discussion

- How long are these reads?
- What is the median quality at the last position?
- View and explore other summary files:
  - G1E_CTCF_summary.txt
  - G1E_input_summary.txt
  - G1E_ER4_CTCF_chr19_summary.txt
  - G1E_ER4_input_chr19_summary.txt

Step 2: Map ChIP-Seq Reads to MM9 Genome

Next, we will map the reads in **G1E_E4R_CTCF_chr9.fastqsanger** to the mouse genome using **Bowtie2.**

Please do not try to Run the commands in the first box. This is just to explain the arguments to bowtie2

usage:

```
bowtie2 [options]    -x <base name of index files> \
                     -U <in_file_name> \
                     -S <out_file_name.sam>
                     --sensitive
```

The index files are available on Biocluster and you do not need to download them now. However it can be downloaded from [Please do not download now]:
ftp://ftp.ccb.jhu.edu/pub/data/bowtie2_indexes/mm9.zip
Script run_bowtie2.sh uses bowtie2 on all four input fastq files and maps them to mm9 genome.

```
$ cd ~/05_Epigenomics/src/

$ sbatch run_bowtie2.sh

# OUTPUT in ~/05_Epigenomics/results/Bowtie_output

$ squeue –u <userID> # to check the status of the submitted job
```

There are other parameters that can be specified for a more controlled use of Bowtie2. In particular, following are some preset options that can be used to modify the speed and sensitivity of the tool:

--very-fast

--fast

--sensitive (default)

--very-sensitive

More information on the Bowtie2 can be found in its well-written manual:

http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (run_bowtie2.sh) is supposed to do in more detail.

What's inside the **run_bowtie2.sh** script?

```
#!/bin/bash
#SBATCH -c 4
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J bowtie2_test
#SBATCH -o bowtie2_test.%j.out
#SBATCH -e bowtie2_test.%j.err
#SBATCH -p classroom
# load the tool environment
module load Bowtie2/
# this is our input (fastq)
export BOWTIE_INP_1=../data/G1E_ER4_CTCF_chr19.fastqsanger
export BOWTIE_INP_2=../data/G1E_ER4_input_chr19.fastqsanger
export BOWTIE_INP_3=../data/G1E_CTCF.fastqsanger
export BOWTIE_INP_4=../data/G1E_input.fastqsanger
mkdir -p ../results/Bowtie_output
# this is our output (SAM)
export BOWTIE_OUT_1=../results/Bowtie_output/G1E_ER4_CTCF_chr19.sam
export BOWTIE_OUT_2=../results/Bowtie_output/G1E_ER4_input_chr19.sam
export BOWTIE_OUT_3=../results/Bowtie_output/G1E_CTCF_chr19.sam
export BOWTIE_OUT_4=../results/Bowtie_output/G1E_input_chr19.sam
# PATH TO the mm9 dm3 index file along with their common prefix (mm9)
export GENOME_MM9_BOWTIE2=../data/index/mm9

bowtie2 -q -x $GENOME_MM9_BOWTIE2 -U $BOWTIE_INP_1 -S $BOWTIE_OUT_1
bowtie2 -q -x $GENOME_MM9_BOWTIE2 -U $BOWTIE_INP_2 -S $BOWTIE_OUT_2
bowtie2 -q -x $GENOME_MM9_BOWTIE2 -U $BOWTIE_INP_3 -S $BOWTIE_OUT_3
bowtie2 -q -x $GENOME_MM9_BOWTIE2 -U $BOWTIE_INP_4 -S $BOWTIE_OUT_4
```

Tells the cluster 'job manager' what resources you want (4 CPUs, 8GB memory, run on the 'classroom' nodes, and name the job 'bowtie2_test'

Load the software. We are using a tool called 'Bowtie2' to Map fastq files to mm9 mouse genome.

Create shortcut names for input and output files, as well as mm9 genome index files' prefix.

Run bowtie2 tool on all four input fastq files

Step 2: Map ChIP-Seq Reads to MM9 Genome

```
$ head -40 ~/05_Epigenomics/results/Bowtie_output/G1E_ER4_CTCF_chr19.sam

# view the first 40 lines of an output SAM file
```

- View and explore other SAM files:
  - G1E_CTCF_chr19.sam
  - G1E_input_chr19.sam
  - G1E_ER4_CTCF_chr19.sam
  - G1E_ER4_input_chr19.sam

You can find more information on the structure of SAM files in the following links:

https://www.samformat.info/sam-format-flag

https://en.wikipedia.org/wiki/SAM_%28file_format%29

Step 3A: Calling Peaks with MACS2

With our mapped ChiP-Seq reads, we now want to call peaks.

We use **MACS2** for this purpose.

usage:

Please do not try to Run the commands in the following box. This is just to explain the arguments to macs2

```
macs2 callpeak     –t <path_to_treatment_input_alignment> \
                   -c < path_to_control_input_alignment > \ # optional
                   -g <effective_genome_size> \ # use mm for mouse
                   -n <prefix_for_naming_output_files>
```

A useful tutorial for MACS2 can be found here:
https://hbctraining.github.io/Intro-to-ChIPseq/lessons/05_peak_calling_macs.html

Step 3A: Calling Peaks with MACS2

Script run_macs2_noControl.sh runs MACS2 to call peaks for G1E_ER4_CTCF_chr19.sam with the default parameters.

Note that this macs2 run is performed without using input from control experiment.

```
$ cd ~/05_Epigenomics/src/

$ sbatch run_macs2_noControl.sh

# OUTPUT in ~/05_Epigenomics/results/MACS2_output/CTCF_ER4_noControl*

$ squeue –u <userID> # to check the status of the submitted job
```

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (run_macs2_noControl.sh) is supposed to do in more detail.

What's inside the **run_macs2_noControl.sh** script?

```
#!/bin/bash
#SBATCH -c 1
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J macs2_noC
#SBATCH -o macs2_noC.%j.out
#SBATCH -e macs2_noC.%j.err
#SBATCH -p classroom
```

Tells the cluster 'job manager' what resources you want (1 CPU, 8GB memory, run on the 'classroom' nodes, and name the job 'macs2_noC'

```
# load the tool environment
module load MACS2/2.1.2-IGB-gcc-4.9.4-Python-2.7.13
```

Load the software. We use a tool called 'MACS2' to call ChIP peaks.

```
# this is our input (SAM)
export MACS_TREAT=../results/Bowtie_output/G1E_ER4_CTCF_chr19.sam

# this is our output_directory
export MACS_OUT_DIR=../results/MACS2_output
# this is our output prefix
export MACS_OUT_1=CTCF_ER4_noControl
```

Create shortcut names for input alignment file, output directory, and output prefix.

```
macs2 callpeak -t $MACS_TREAT  -g mm -f SAM --outdir $MACS_OUT_DIR -n $MACS_OUT_1
```

Run MACS2 tool on G1E_ER4_CTCF_chr19.sam without using the control experiment

Step 3A: Calling Peaks with MACS2

Number of peaks called without using a control experiment input can be obtained using:

```
$ cat ../results/MACS2_output/CTCF_ER4_noControl_peaks.narrowPeak | wc -l

# You should get 626

$ head ../results/MACS2_output/CTCF_ER4_noControl_peaks.narrowPeak
```

Here are the fields (columns) of a .narrowPeak file:

1. chromosome
2. start coordinate
3. end coordinate
4. name
5. score
6. strand
7. **signalValue - Measurement of overall enrichment for the region**
8. **pValue - Statistical significance (-log10)**
9. **qValue - Statistical significance using false discovery rate (-log10)**
10. **peak - Point-source called for this peak; 0-based offset from chromStart**

Standard BED file fields

narrowPeak specific fields

screenshot from:
https://hbctraining.github.io/Intro-to-ChIPseq/lessons/05_peak_calling_macs.html

Call ChIP-Seq Peaks with a Control Sample

We will perform the same procedure we did in the previous exercise. This time though, we will work with a control sample in addition to the treated one.

Step 3B: Calling Peaks with MACS2 using Control Chip-Seq Reads

Script run_macs2_noControl.sh runs MACS2 to call peaks for G1E_ER4_CTCF_chr19.sam with the default parameters.

Note that this macs2 run is performed using additional input from control experiment (G1E_ER4_input_chr19.sam).

```
$ cd ~/05_Epigenomics/src/

$ sbatch run_macs2_withControl.sh

# OUTPUT in ~/05_Epigenomics/results/MACS2_output/CTCF_ER4_withControl*

$ squeue –u <userID> # to check the status of the submitted job
```

Number of peaks called using the additional control experiment input can be obtained using:

```
$ cat ../results/MACS2_output/CTCF_ER4_withControl_peaks.narrowPeak | wc -l

# You should get 528
```

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (run_macs2_withControl.sh) is supposed to do in more detail.

What's inside the **run_macs2_withControl.sh** script?

```
#!/bin/bash
#SBATCH -c 1
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J macs2_wC
#SBATCH -o macs2_wC.%j.out
#SBATCH -e macs2_wC.%j.err
#SBATCH -p classroom

# load the tool environment
module load MACS2/2.1.2-IGB-gcc-4.9.4-Python-2.7.13

# this is our input (SAM)
export MACS_TREAT=../results/Bowtie_output/G1E_ER4_CTCF_chr19.sam
export MACS_CONTROL=../results/Bowtie_output/G1E_ER4_input_chr19.sam
# this is our output directory
export MACS_OUT_DIR=../results/MACS2_output
# this is our output prefix
export MACS_OUT_1=CTCF_ER4_withControl


macs2 callpeak -t $MACS_TREAT -c $MACS_CONTROL -g mm -f SAM –outdir \
$MACS_OUT_DIR  -n $MACS_OUT_1
```

Tells the cluster 'job manager' what resources you want (1 CPU, 8GB memory, run on the 'classroom' nodes, and name the job 'macs2_wC'

Load the software. We use a tool called 'MACS2' to call ChIP peaks.

Create shortcut names for treatment and control input alignment files, output directory, and output prefix.

Run MACS2 tool on G1E_ER4_CTCF_chr19.sam while using G1E_ER4_input_chr19.sam as the control experiment.

Step 3C: Calling Peaks with MACS2 on Chip-Seq Reads for un-stimulated cells

Script run_macs2_noER.sh runs MACS2 to call peaks for G1E_CTCF_chr19.sam with the default parameters.

Note that this macs2 run is performed using additional input from control experiment (G1E_input_chr19.sam).

```
$ cd ~/05_Epigenomics/src/

$ sbatch run_macs2_noER.sh

# OUTPUT in ~/05_Epigenomics/results/MACS2_output/CTCF_noE2_*

$ squeue –u <userID> # to check the status of the submitted job
```

Exercise:
Find out the number of peaks called for this ChIP-Seq experiment

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (run_macs2_noER.sh) is supposed to do in more detail.

What's inside the **run_macs2_noER.sh** script?

```
#!/bin/bash
#SBATCH -c 4
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J macs2_noER
#SBATCH -o macs2_noER.%j.out
#SBATCH -e macs2_noER.%j.err
#SBATCH -p classroom

# load the tool environment
module load MACS2/2.1.2-IGB-gcc-4.9.4-Python-2.7.13

# this is our input (SAM)
export MACS_TREAT_NOER=../results/Bowtie_output/G1E_CTCF_chr19.sam
export MACS_CONTROL_NOER=../results/Bowtie_output/G1E_input_chr19.sam
# this is the output directory
export MACS_OUT_DIR=../results/MACS2_output
# this is our output prefix
export MACS_OUT_1=CTCF_noE2


macs2 callpeak -t $MACS_TREAT_NOER -c $MACS_CONTROL_NOER -g mm -f SAM –outdir \
$MACS_OUT_DIR  -n $MACS_OUT_1
```

Tells the cluster 'job manager' what resources you want (1 CPU, 8GB memory, run on the 'classroom' nodes, and name the job 'macs2_noER'

Load the software. We use a tool called 'MACS2' to call ChIP peaks.

Create shortcut names for treatment and control input alignment files, output directory, and output prefix.

Run MACS2 tool on G1E_CTCF_chr19.sam while using G1E_input_chr19.sam as the control experiment.

MACS2 summary

MACS2 creates three output files:
_peaks.narrowPeak: BED6+4 format file which contains the peak locations together with peak summit, pvalue and qvalue.
_peaks.xls: a tabular file which contains information about called peaks. Additional information includes pileup and fold enrichment

Discussion

1. Examine the **BED** tracks.
2. How many peaks are called when using a control sample?
3. How does this compare to the previous situation where we only had experimental Chip-Seq reads?

Identifying Differential Binding Sites

In this exercise, we will identify binding sites exclusive to undifferentiated and differentiated cell lines as well as those common to both, using “bedtools” toolkit.

Step 4A: Subtract Peaks Between Cell Lines

we will use “bedtools intersect” tool from bedtools toolkit to identify CTCF peaks that are unique to the differentiated cell line:

Usage:

Please do not try to Run the commands in the following box. This is just to explain the arguments to bedtools

```
$ bedtools intersect[options]  -a <first_interval.bed> \

                               -b <second_interval.bed>
```

Specific options determine the behaviour of bedtools intersect, e.g.

-wa Write the original entry in A for each overlap.

-wb Write the original entry in B for each overlap.

-v Only report those entries in A that have no overlap in B.

As an example, following command finds entries in A.bed that are absent in B.bed:

Please do not try to Run the commands in the following box. This is just to show an example.

```
$ bedtools intersect –v –a A.bed –b B.bed
```

Here is a link to the manual for bedtools intersect:
https://bedtools.readthedocs.io/en/latest/content/tools/intersect.html

Step 4A: Subtract Peaks Between Cell Lines.

Use the following command to find peaks in E2 treated cells that are absent in untreated cells:

```
$ cd ~/05_Epigenomics/src/

$ sbatch bedtools_subtract_1.sh

# OUTPUT in ~/05_Epigenomics/results/peak_inspection/CTCF_subtract_1.bed

$ squeue –u <userID> # to check the status of the submitted job
```

The resulting **BED** file (CTCF_subtract_1.bed) contains peaks exclusive to the **differentiated** cell line (G1E-ER4).

Discussion

1. How many peaks are exclusive to G1E-ER4?

```
$ cat ~/05_Epigenomics/results/peak_inspection/CTCF_subtract_1.bed | wc -l

# You should get 136
```

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (bedtools_subtract_1.sh) is supposed to do in more detail.

What's inside the **bedtools_subtract_1.sh** script?

```
#!/bin/bash
#SBATCH -c 1
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J bedtools_subt1
#SBATCH -o bedtools_subt1.%j.out
#SBATCH -e bedtools_subt1.%j.err
#SBATCH -p classroom

# load the tool environment
module load BEDTools

# this is our input (bed like)
export PEAK_1=../results/MACS2_output/CTCF_ER4_withControl_peaks.narrowPeak
export PEAK_2=../results/MACS2_output/CTCF_noE2_peaks.narrowPeak

mkdir -p ../results/peak_inspection
# this is our output (bed like)
export OUT_1=../results/peak_inspection/CTCF_subtract_1.bed


bedtools intersect -v -a $PEAK_1 -b $PEAK_2 > $OUT_1
```

Tells the cluster 'job manager' what resources you want (1 CPU, 8GB memory, run on the 'classroom' nodes, and name the job 'bedtools_subt1'

Load the software. We use a tool called 'BEDTools' to work with generated peak files.

Create shortcut names for the two input bed files, and the output bed file.

run 'bedtools intersect' using the –v flag to get the difference between the two bed files, and store the results in CTCF_subtract_1.bed

Step 4A: Subtract Peaks Between Cell Lines

Redo Step1 only **SWITCH** the input order to get the peaks unique to the untreated cells.

use the following command to do just that:

```
$ cd ~/05_Epigenomics/src/

$ sbatch bedtools_subtract_2.sh

# OUTPUT in ~/05_Epigenomics/results/peak_inspection/CTCF_subtract_2.bed
```

The resulting **BED** file (CTCF_subtract_2.bed) contains peaks exclusive to the **undifferentiated** cell line (G1E).

Exercise:
How many peaks are exclusive to the undifferentiated cell line?

```
$ cat ~/05_Epigenomics/results/peak_inspection/CTCF_subtract_2.bed | wc -l

# You should get 23
```

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (bedtools_subtract_2.sh) is supposed to do in more detail.

What's inside the **bedtools_subtract_2.sh** script?

```
#!/bin/bash
#SBATCH -c 1
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J bedtools_subt2
#SBATCH -o bedtools_subt2.%j.out
#SBATCH -e bedtools_subt2.%j.err
#SBATCH -p classroom

# load the tool environment
module load BEDTools

# this is our input (bed like)
export PEAK_1=../results/MACS2_output/CTCF_ER4_withControl_peaks.narrowPeak
export PEAK_2=../results/MACS2_output/CTCF_noE2_peaks.narrowPeak

mkdir -p ../results/peak_inspection
# this is our output (bed like)
export OUT_1=../results/peak_inspection/CTCF_subtract_2.bed


bedtools intersect -v -a $PEAK_2 -b $PEAK_1 > $OUT_1
```

Tells the cluster 'job manager' what resources you want (1 CPU, 8GB memory, run on the 'classroom' nodes, and name the job 'bedtools_subt2'

Load the software. We use a tool called 'BEDTools' to work with generated peak files.

Create shortcut names for the two input bed files, and the output bed file.

run 'bedtools intersect' using the –v flag to get the difference between the two bed files, and store the results in CTCF_subtract_2.bed

Step 4B: Intersect Peaks Between Cell Lines

Following command finds entries in A.bed that overlap with at least one entry in B.bed:

Please do not try to Run the command in the following box. This is just to show an example.

```
$ bedtools intersect –wa –a A.bed –b B.bed
```

Use the following command to find peaks in E2 treated cells that overlap with peaks in the untreated cells:

```
$ cd ~/05_Epigenomics/src/

$ sbatch bedtools_overlap_1.sh

# OUTPUT in ~/05_Epigenomics/results/peak_inspection/CTCF_overlap_1.bed
```

The resulting **BED** file (CTCF_overlap_1.bed) contains peaks from the differentiated cell line (G1E_ER4) that overlap with peaks in the undifferentiated cell line (G1E).

Please do not try to Run the commands in this slide. This is just to explain what the script that we just ran (bedtools_overlap_1.sh) is supposed to do in more detail.

What's inside the **bedtools_overlap_1.sh** script?

```
#!/bin/bash
#SBATCH -c 4
#SBATCH --mem 8000
#SBATCH -A Mayo_Workshop
#SBATCH -J bedtools_ovl1
#SBATCH -o bedtools_ovl1.%j.out
#SBATCH -e bedtools_ovl1.%j.err
#SBATCH -p classroom

# load the tool environment
module load BEDTools

# this is our input (bed like)
export PEAK_1=../results/MACS2_output/CTCF_ER4_withControl_peaks.narrowPeak
export PEAK_2=../results/MACS2_output/CTCF_noE2_peaks.narrowPeak
mkdir -p ../results/peak_inspection
# this is our output (bed like)
export OUT_1=../results/peak_inspection/CTCF_overlap_1.bed

bedtools intersect -wa -a $PEAK_1 -b $PEAK_2 > $OUT_1
```

Tells the cluster 'job manager' what resources you want (1 CPU, 8GB memory, run on the 'classroom' nodes, and name the job 'bedtools_ovl1'

Load the software. We use a tool called 'BEDTools' to work with generated peak files.

Create shortcut names for the two input bed files, and the output bed file.

run 'bedtools intersect' using the –wa flag to get the entries in the first file (-a) that have an overlap with an entry in second input file (-b) and store the results in CTCF_overlap_1.bed