Epigenomic Analysis 2023

Faculty: Guillaume Bourque, Martin Hirst, David Bujold, Jose Hector Galvez, Edmund Su, Mare

October 11, 2023 - October 13, 2023

Contents

Part I

Introduction

Chapter 1

Workshop Info

Welcome to the 2023 Epigenomic Analysis Canadian Bioinformatics Workshop webpage!

1.1 Pre-work

You can find your pre-work here.

1.2 Class Photo

1.3 Schedule

Chapter 2

Meet Your Faculty

2.0.0.1 Guillaume Bourque

Professor, McGill University Director of Bioinformatics, Genome Quebec Innovation Centre Director, Canadian Centre of Computational Genomics Director, McGill initiative for Computational Medicine

Dr. Bourque research interests are in comparative and functional genomics with a special emphasis on applications of next-generation sequencing technologies. His lab develops advanced tools and scalable computational infrastructure to enable large-scale applied research projects.

2.0.0.2 Martin Hirst

Distinguished Scientist, BC Cancer Professor, Department of Microbiology & Immunology Director, Michael Smith Laboratories Michael Smith Laboratories

Dr. Hirst's research focuses on understanding epigenetic dysfunction in cancer and his laboratory develops experimental and computational tools to characterize normal and transformed cell types down to the single cell level. He applies these tools to explore the epigenomic states of normal and transformed cell types to discover and exploit therapeutic vulnerabilities.

2.0.0.3 David Bujold

Bioinformatics Manager, Data Unit Canadian Centre of Computational Genomics

He joined the McGill Epigenomic Data Coordination Center at McGill in 2012 to tackle challenges related to epigenomics, and has since developed many data management and discovery solutions, including the IHEC Data Portal. Other projects of interest include CanDIG and EpiShare, platforms to make genomic and epigenomic data under controlled access more accessible, while maintaining study participants' privacy.

2.0.0.4 Jose Hector Galvez

Bioinformatics Manager, Tech Dev Unit Canadian Centre of Computational Genomics

As a Bioinformatics Specialist in the Research and Development team, Jose Hector is involved in maintaining, documenting, and upgrading the RNA-seq pipelines in GenPipes. He also collaborates in several research projects, mostly focusing on transcriptomics, genome assembly, and epigenomics.

2.0.0.5 Edmund Su

Bioinformatician Ontario Institute for Cancer Research

Edmund is a bioinformatician within the genome informatics team at OICR, where he provides technical knowledge and expertise in genomic analysis. His main focus is developing pipelines and data wrangling for ICGC-ARGO (International Cancer Genome Consortium - Accelerating Research in Genomic Oncology).

2.0.0.6 Mareike Janiak

Bioinformatics Analyst, Tech Dev Unit Canadian Centre of Computational Genomics

As a Bioinformatics Analyst in the TechDev team, Mareike is responsible for maintaining pipelines in GenPipes, testing and developing new pipelines for the community, and responding to user requests. Prior to joining C3G, she worked as both a field and computational biologist, with her doctoral and postdoctoral research focusing on mammalian comparative genomics, phylogenomics, and metagenomics, most often in primates.

2.0.0.7 Michelle Brazas

Acting Scientific Director Canadian Bioinformatics Workshops (CBW) Toronto, ON, CA

— support@bioinformatics.ca

Dr. Michelle Brazas is the Associate Director for Adaptive Oncology at the Ontario Institute for Cancer Research (OICR), and acting Scientific Director at Bioinformatics.ca. Previously, Dr. Brazas was the Program Manager for Bioinformatics.ca and a faculty member in Biotechnology at BCIT. Michelle co-founded and runs the Toronto Bioinformatics User Group (TorBUG) now in its 11th season, and plays an active role in the International Society of Computational Biology where she sits on the Board of Directors and Executive Board.

2.0.0.8 Nia Hughes

Program Manager, Bioinformatics.ca Ontario Institute for Cancer Research Toronto, ON, Canada

— nia.hughes@oicr.on.ca

Nia is the Program Manager for Bioinformatics.ca, where she coordinates the Canadian Bioinformatics Workshop Series. Prior to starting at OICR, she completed her M.Sc. in Bioinformatics from the University of Guelph in 2020 before working there as a bioinformatician studying epigenetic and transcriptomic patterns across maize varieties.

2.0.0.9 Zhibin Lu

HPC and Bioinformatics Services Manager at Princess Margaret Cancer Centre, University Health Network Bioinformatics and HPC Core, UHN MaRS Centre, PMCRT 11-707 101 College St Toronto ON M5G 1L7

— zhibin@gmail.com https://bhpc.uhnresearch.ca/

Zhibin Lu is a senior manager at University Health Network Digital. He is responsible for UHN HPC operations and scientific software. He manages two HPC clusters at UHN, including system administration, user management, and maintenance of bioinformatics tools for HPC4health. He is also skilled in Next-Gen sequence data analysis and has developed and

maintained bioinformatics pipelines at the Bioinformatics and HPC Core. He is a member of the Digital Research Alliance of Canada Bioinformatics National Team and Scheduling National Team.

Chapter 3

Data and Compute Setup

3.0.0.1 Course data downloads

Some of these files are quite large. If you have issues downloading them, contact us at support@bioinformatics.ca.

- Module 123
- Module 4

3.0.0.2 Compute setup

3.0.0.2.1 AWS Module Connecting and properly using a cloud computing cluster at the CBW here

3.0.0.2.2 AMI We have made our AWS AMI (Amazon Machine Image) publicly available. To launch your own instance, follow the instructions provided by Amazon on how to launch an EC2 instance from a custom Amazon Machine Image. Please note that you will need an AWS account to proceed, and that you will need to upload the CourseData files yourself.

Here are the details of the AMI:

- AWS Region: us-east-1 (N. Virgina)
- AMI type: public image
- AMI name: CBW_EPI_231005
- AMI ID: ami-0ae28d3f8e57ba8cf

If you want to create and activate a new AWS account, please follow the instructions provided by Amazon.

3.0.0.2.3 Software

3.0.0.2.3.1 conda Install Miniconda by following the instructoion at Miniconda official site

3.0.0.2.3.2 bedtools

```
wget https://github.com/arq5x/bedtools2/releases/download/v2.30.0/bedtools.static.binar
mv bedtools.static.binary bedtools
chmod +x bedtools
```

3.0.0.2.3.3 bwa

```
wget https://newcontinuum.dl.sourceforge.net/project/bio-bwa/bwa-0.7.17.tar.bz2
tar -jxvf bwa-0.7.17.tar.bz2
cd bwa-0.7.17/
```

3.0.0.2.3.4 make

```
#### FastQC
wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.12.1.zip
unzip fastqc_v0.12.1.zip
```

3.0.0.2.3.5 samtools

```
wget https://github.com/samtools/samtools/releases/download/1.17/samtools-1.17.tar.bz2
tar -jxvf samtools-1.17.tar.bz2
cd samtools-1.17
make
sudo make install
```

3.0.0.2.3.6 picard

```
wget https://github.com/broadinstitute/picard/releases/download/3.1.0/picard.jar
```

3.0.0.2.3.7 deeptools

```
pip install deeptools
```

3.0.0.2.3.8 MACS2

```
pip install MACS2
```

3.0.0.2.3.9 Homer

```
wget http://homer.ucsd.edu/homer/configureHomer.pl
perl ./configureHomer.pl -install
perl ./configureHomer.pl -install hg19
perl ./configureHomer.pl -install hg38
```

3.0.0.2.3.10 UCSC tools'

```
rsync -aP hgdownload.soe.ucsc.edu::genome/admin/exe/linux.x86_64/ ./
```

3.0.0.2.3.11 Biamark

```
wget https://github.com/FelixKrueger/Bismark/archive/refs/tags/v0.24.2.tar.gz
tar -zxf v0.24.2.tar.gz
```

Part II

Modules

Chapter 4

Module 1: Introduction to ChIP Sequencing and Analysis

4.1 Lecture

4.2 Lab

4.2.1 Breakdown of Markdown File

- At the start of each step, the intention will declare.
- This is then follow by a code block

Code:

```
Like this! This is the main code to run for the step.

Additionally the code block will include a header to indicate what environment to the run code fo
###Shell###
pwd

###R###
getwd()
```

- Explaining for commands will be broken down following code block
- `pwd` & `getwd()`

  - See your work directory
- Sprinkled throughout will also include comments

Important points and considerations will also be raised as so.

4.2.2 Module 1. Alignment

4.2.2.1 Step 1: Setup

- It's always good practice to organize your directories beforehand
- This avoids file conflicts and accidental data loss.
- We'll be making a subdirectory for each major step.

Code:

```
###Shell###
mkdir -p ~/workspace/module123/BWA_index
mkdir -p ~/workspace/module123/alignments
mkdir -p ~/workspace/module123/fastqc
mkdir -p ~/workspace/module123/stats
mkdir -p ~/workspace/module123/peaks
mkdir -p ~/workspace/module123/bigBed
mkdir -p ~/workspace/module123/bigWig
mkdir -p ~/workspace/module123/resources
mkdir -p ~/workspace/module123/diffBind
mkdir -p ~/workspace/module123/edgeR
mkdir -p ~/workspace/module123/qc
```

- `mkdir -p ~/workspace/module123/BWA_index`
  - `mkdir` creates a directory
  - `-p` creates the "parent" if the initial parent directory did not exist
  - how could we simplfy the command?
  - `mkdir -p ~/workspace/module123/{BWA_index,alignments,fastqc,stats,peaks,bigBe`

4.2.2.2 Step 2A: Retrieve a reference genome

- We'll need a reference genome to align our sequences to.
- A great resource is UCSC's genome repository containing latest `Hg38` and older `hg19` human genome.
- For our tutorial, we'll be working `chr19` of `Hg38`/`Grch38`. This is to speed demonstration purposes.

Code:

```
###Shell###
wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr19.fa.gz -O ~/workspace/modul
gunzip ~/workspace/module123/BWA_index/chr19.fa.gz -c > ~/workspace/module123/BWA_index/chr19.fa
```

- Runtime : <1 minute
- `wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr19.fa.gz -O ~/workspace/module123/BWA_index/chr19.fa.gz`
  - `Wget` is a command to pull online files.
  - We're instructing it to pull `chr19.fa.gz` from the example address.
  - `-O ~/workspace/module123/BWA_index/chr19.fa.gz` instructs where and what we would like to save our file as.
  - *What were to happen if we ran the command without `-O`?*
  - Not all systems carry/support `wget`, `Curl` is another useful alternative.
- `gunzip ~/workspace/module123/BWA_index/chr19.fa.gz -c > ~/workspace/module123/BWA_index/chr19.fa`
  - `gzip` is a data compression format useful for reducing storage impacts
  - `gunzip` is the opposite decompresses `gzipped` data
  - `-c` is a STDOUT redirect. Normally running `gunzip file.gz` will turn `file.tsv.gz` into `file.tsv`. Running `-c` allows us to save the contents elsewhere. We do this we can compare the size difference.
  - compare `ls ~/workspace/module123/BWA_index/chr19.fa.gz -ilth` vs `ls ~/workspace/module123/BWA_index/chr19.fa -ilth` and *note the size difference*

Prior to any tool usage, it's good practice to explore the tool by invoking the help command. This may vary depending on tool. This is important as the same `-O` flag could have differing functions in different tools.

4.2.2.3 Step 2B: Index the genome

- Ready the genome fasta file by creating databases allowing tools to quickly access different parts of the file
- Indexing is only required once per genome.
- That being said, because there are different versions of `Hg38`/`GRCh38` (such as those with or without ALT contigs and EBV) each version would need their own indices.
- Additionally index files are typically program/tool specific.
- Consortias will also have available the reference genome and related resources. E.g. https://ewels.github.io/AWS-iGenomes/

A newer version of BWA-mem does exist, however we'll be using the older version due to a bug that requires significant memory for indexing.

Code:

```
###Shell###
mkdir -p ~/workspace/module123/BWA_index
bwa index ~/workspace/module123/BWA_index/chr19.fa > ~/workspace/module123/BWA_index/in
```

- Runtime : <1 minute
- `bwa index ~/workspace/module123/BWA_index/chr19.fa`
  - Indexing step creates a database to enable quick retrieve of sequences
  - take a look inside your folder `ls ~/workspace/module123/BWA_index/`
  - For the lab, we're utilizing BWA for alignment. ### Step 3A: FastQC and interpretation
- Checking the quality of your FASTQs is sanity check step (garbage in garbage out)

Code:

```
###Shell###
fastq_input=~/CourseData/EPI_data/module123/fastq/MCF10A.Input.chr19.R1.fastq.gz
fastq_h3k4me3=~/CourseData/EPI_data/module123/fastq/MCF10A.H3K4me3.chr19.R1.fastq.gz
mkdir -p ~/workspace/module123/fastqc
fastqc -t 8 ${fastq_input} -o ~/workspace/module123/fastqc > ~/workspace/module123/fast
fastqc -t 8 ${fastq_h3k4me3} -o ~/workspace/module123/fastqc > ~/workspace/module123/fa
```

- `fastqc`
  - `-t 8` specifies the number of threads we want to program to utilize
  - `-o` specifies our output directory ### Step 3A: FastQC and interpretation
- Let's take a look at our FASTQC results and compare

Code:

```
###Browser###
http://main.uhn-hpc.ca/module123/fastqc/MCF10A.Input.chr19.R1_fastqc.html
http://main.uhn-hpc.ca/module123/fastqc/MCF10A.H3K4me3.chr19.R1_fastqc.html
```

The URLs link to the TA's instance, make sure to replace the domain with your own.

- Observe the summary report, note the warnings and errors.
- we can also look at reports at https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
- `Per base sequence quality`
  - a representation of quality across all reads where the ideal is average,median,25th and 75th resides within the green good quality space
  - X-axis is the BP position in your read
  - Y-axis is the base quality score
  - yellow boxes represent 25th and 75th percentiles
  - whiskers represent 10th and 90th percentiles
  - read line represents median
  - blue line represents average
  - For illumina reads, the first couple of bases tend to be poor, quality normalizes for majority of the read and slowly worsens approaching end of read
  - possible remedies are to trim reads
- `Per tile sequence quality`
  - a representation of quality by tile (a discrete subdivision with the lane of a flowcell) where heat is bad
  - would indicate an issue with the flowcell
- `Per sequence quality score`
  - a distribution of mean read quality score where the peak and body should be as far right as possible
- `Per base sequence content`
  - The % of AGCT across reads according to read position
  - generally expect an equal/even except for the first couple of base pairs (due to sequencing calibration)
  - Certain assays would change this for example : Bisulphite-Sequencing (unmethylated Cs become Us) or amplicon-sequencing (fixed ratios)
  - look at H3K4me3, why is it so high in GC? Promoters!
- `Per sequence GC content`
  - a distrubtion of reads' GC content, where warning or failures are issued based on deviation of 15% or 30%
  - A shift in distribution could mean sequencing bias or contamination
- `Per base N content`
  - B/C `N`s are low confidence calls by the sequencer, we expect very few instances
- `Sequence Length Distribution`
  - With Illumina sequencing, expect uniform distribution

- `Sequence duplication levels`
  - Using the first 200,000 to find duplicates: expect most reads to unique (left side), spikes otherwise indicate contamination or PCR artifacts
- `Overrepresented sequences`
  - a table breakdown of the sequences found above in `Sequence duplication levels`
- `Adapter Content`
  - detects if commonly used adapters remain in read sequence

4.2.2.4 Step 4: Alignment

- Mapping the read pairs to a position of the reference genome

Code:

```
###Shell###
ref=~/workspace/module123/BWA_index/chr19.fa
read1=~/CourseData/EPI_data/module123/fastq/MCF10A.Input.chr19.R1.fastq.gz
read2=~/CourseData/EPI_data/module123/fastq/MCF10A.Input.chr19.R2.fastq.gz
sample=MCF10A_input_chr19
bwa mem -M -t 4 ${ref} ${read1} ${read2} 2>~/workspace/module123/alignments/${sample}.a
```

- run time ~2 min
- The command can be broken down into the following pseudo code `ALIGNMENT | SAMtoBAM_conversion`. The `|` operate streams the results from the first command into the second.
- `bwa mem -M -t 4 ${ref} ${read1} ${read2} 2>~/workspace/module123/alignments/alignm`
  - `*` `|` the pipe delimiter passes the results to the next step `ACTION A | ACTION B | ACTION C` (like an assembly line)
  - `bwa mem` while BWA has many alignment algorithms, `mem` is best suited for efficiently handling >70bp paired end reads.
  - `-M` alters flagging of chimeric reads. By default chimeric reads are flagged as `SUPPLEMENTARY` (partially mapping), the option instead turns them to `SECONDARY` (multi-mapping). Needed to GATK/PICARD support downstream
  - `-t 4` resource specification
  - `2>~/workspace/module123/alignments/alignment.log` data outputted occurs in two streams `1` (the data) and `2` (debug messages, warnings and status updates). We redirect `2` to a log file.
  - What do we see in the logs?

Logs contain valuable information, helpful for debugging.

- `samtools view -hbS -o ~/workspace/module123/alignments/${sample}.bam`
  - `samtools view` a tool to read our `SAM`,`BAM`,`CRAM` files
  - `-hbS` include header, output as `BAM`, input is `SAM`
  - `-o` specify what we want to save our file as
  - take a look our new `BAM` file. Note the header and the body.
- What other important functions of `samtools`?
  - CRAM conversion
  - samtools quickcheck
  - samtools view header
  - samtools flags ### Step 5: Coordinate Sort
- Rearrange our alignments by coordinates

Code:

```
###Shell###
sample=MCF10A_input_chr19
samtools sort -@8 ~/workspace/module123/alignments/${sample}.bam -o ~/workspace/module123/alignme
```

- run time <1 min
- `samtools sort` sorts our reads by genome coordinates
- Observe the files before and after via `samtools view | head`
- How to sort by readname? #### Step 6: Duplicate Marking
- Identify and tag alignments that are duplicates

Code:

```
###Shell###
sample=MCF10A_input_chr19
java -jar /usr/local/picard.jar MarkDuplicates \
I=~/workspace/module123/alignments/${sample}.sorted.bam \
O=~/workspace/module123/alignments/${sample}.sorted.dup_marked.bam \
M=~/workspace/module123/alignments/${sample}.dup_marked.output.log \
ASSUME_SORTED=TRUE \
VALIDATION_STRINGENCY=LENIENT \
> ~/workspace/module123/alignments/${sample}.dup_marked.error.log
```

- `java -jar /usr/local/picard.jar MarkDuplicates I=~/workspace/module123/alignments/${sample}.`
  `O=~/workspace/module123/alignments/${sample}.sorted.dup_marked.bam`
  `M=~/workspace/module123/alignments/${sample}.dup_marked.output.log`
  `ASSUME_SORTED=TRUE VALIDATION_STRINGENCY=LENIENT`

  - `java -jar /usr/local/picard.jar` produced by BroadInstitute, Picard tools are a toolset for HTS/NGS data
  - `MarkDuplicates` identifies reads/clusters duplicates arising from library construciton during PCR and cluster formation during sequencing
  - `I=` and `O=` are input and output respectively
  - `M=` saves our output log
  - `ASSUME_SORTED=TRUE` informs `PICARD`, our input is already coordinate sorted
  - `VALIDATION_STRINGENCY=LENIENT` informs `PICARD` to continue and notify problems in the `work.log` instead of failing

4.2.2.5 Step 7: Stats

- Identify and tag alignments that are duplicates

Code:

```
###Shell###
sample=MCF10A_input_chr19
mkdir -p  ~/workspace/module123/stats
samtools flagstat ~/workspace/module123/alignments/${sample}.sorted.dup_marked.bam > ~/
samtools stats ~/workspace/module123/alignments/${sample}.sorted.dup_marked.bam > ~/wo
```

- `samtools flagstat` tallies `FLAGS` and produces a summarized report
- `samtools stats` collects various metrics on the BAM file include:
  - summary stats similar to `flagstats`
  - distribution of insert sizes
  - distribution of read lengths
  - distribution of Coverage
  - distribution of GC-depth
  - Includes detailed instructions on how to extract parts of interest

4.2.2.6 Step 8: Processing the remaining files

- we use a unix for loop to perform all the steps previously mentioned but for the other histone marks and ATAC-seq **Code:**

```
###Shell##
ref=~/workspace/module123/BWA_index/chr19.fa
for histone in H3K27ac H3K27me3 H3K4me3 ATAC;
    do
    read1=~/CourseData/EPI_data/module123/fastq/MCF10A.${histone}.chr19.R1.fastq.gz
```

```
read2=~/CourseData/EPI_data/module123/fastq/MCF10A.${histone}.chr19.R2.fastq.gz
sample=MCF10A_${histone}_chr19
echo "aligning ${histone}"
bwa mem -M -t 4 ${ref} ${read1} ${read2} 2> ~/workspace/module123/alignments/${sample}.alignm
echo "sorting ${histone}"
samtools sort -@8 ~/workspace/module123/alignments/${sample}.bam -o ~/workspace/module123/ali
echo "dupmarking ${histone}"
java -jar /usr/local/picard.jar MarkDuplicates I=~/workspace/module123/alignments/${sample}.s
echo "calculating stats ${histone}"
samtools flagstat ~/workspace/module123/alignments/${sample}.sorted.dup_marked.bam > ~/worksp
samtools stats ~/workspace/module123/alignments/${sample}.sorted.dup_marked.bam > ~/workspace
done
```

- run time ~8 mins
- for loops are different in every language, best practice to review beforehand
- have feedback to indicate where in the process we are e.g. the echo lines
- not show here but also good to capture success and failures

4.2.2.7 Step 9: Clean up!

- Remove temporary files

Code:

```
###Shell###
ls ~/workspace/module123/alignments/*.bam | grep -v sorted.dup_marked | xargs -I {} sh -c "echo r
```

- `ls ~/workspace/module123/alignments/*.bam`
  - a look up of `BAM` files in our directory : `~/workspace/module123/alignments`
  - The `*` in `*.bam` acts as a wild card for any string of variable length ending in the suffix `bam`
  - running the command on it's own produces
- `grep -v sorted.dup_marked`
  - `grep` a powerful tool that searches and matches text
  - `-v` return matches that do not much our pattern or regex
  - `sorted.dup_marked` is our pattern, returning 2/3 of the files : `MCF10A_input_chr19.bam,MCF10A_input_chr19.sorted.bam,MCF10A_input_chr19.sorted.dup_marked`
- `xargs -I {} sh -c "echo rm {}; rm {}"`
  - `xargs` receives input and performs an action. Think a for-loop
  - `-I {}` the variable we want to use
  - `sh -c` interpret the string provided in bash shell
  - `echo rm {}` echos the command we want to perform
  - `rm {}` removes the file

4.2.3 Server resources

4.2.3.1 QC Resources

```
###TSS+/-2kb
mkdir workspace/module123/qc
wget https://www.bcgsc.ca/downloads/esu/touchdown/hg38v79_genes_tss_2000.bed -O workspa

sort -k1,1 -k2,2n workspace/module123/resources/hg38v79_genes_tss_2000.bed > tmp
mv tmp workspace/module123/resources/hg38v79_genes_tss_2000.bed

###Enhancer liftover
wget https://www.bcgsc.ca/downloads/esu/touchdown/encode_enhancers_liftover.bed -O worl

###Blacklist
wget https://www.encodeproject.org/files/ENCFF356LFX/@@download/ENCFF356LFX.bed.gz -O

gunzip ~/workspace/module123/resources/hg38_blacklist.bed.gz
```

- `hg38v79_genes_tss_2000.bed`
  - Generated by downloading Ensemblv79 GTF convert to Bed +/-2kb of TSS. See https://www.biostars.org/p/56280/
- `encode_enhancers_liftover.bed`
  - download various ChroHMM state7 and merge
  - converted from hg19 to hg38 using UCSC liftover tool

4.2.3.2 Encode Bed

```
ls ~/CourseData/EPI_data/module123/encode_bed
basal.H3K27ac.peak_calls.bed
basal.H3K27me3.peak_calls.bed
basal.H3K4me1.peak_calls.bed
basal.H3K4me3.peak_calls.bed
lp.H3K27ac.peak_calls.bed
lp.H3K27me3.peak_calls.bed
lp.H3K4me1.peak_calls.bed
lp.H3K4me3.peak_calls.bed
luminal.H3K27ac.peak_calls.bed
luminal.H3K27me3.peak_calls.bed
luminal.H3K4me1.peak_calls.bed
luminal.H3K4me3.peak_calls.bed
stromal.H3K27ac.peak_calls.bed
stromal.H3K27me3.peak_calls.bed
```

```
stromal.H3K4me1.peak_calls.bed
stromal.H3K4me3.peak_calls.bed
```

- https://epigenomesportal.ca/tracks/CEEHRC/hg38/
- Breast Basal CEMT0035
- Breast Stromal CEMT0036
- Breast Luminal CEMT0037
- Breast Luminal Progenitor CEMT0038 #### Encode BigWig

```
ls ~/CourseData/EPI_data/module123/encode_bigWig
basal.H3K27ac.signal_unstranded.bigWig
basal.H3K27me3.signal_unstranded.bigWig
basal.H3K4me1.signal_unstranded.bigWig
basal.H3K4me3.signal_unstranded.bigWig
lp.H3K27ac.signal_unstranded.bigWig
lp.H3K27me3.signal_unstranded.bigWig
lp.H3K4me1.signal_unstranded.bigWig
lp.H3K4me3.signal_unstranded.bigWig
luminal.H3K27ac.signal_unstranded.bigWig
luminal.H3K27me3.signal_unstranded.bigWig
luminal.H3K4me1.signal_unstranded.bigWig
luminal.H3K4me3.signal_unstranded.bigWig
stromal.H3K27ac.signal_unstranded.bigWig
stromal.H3K27me3.signal_unstranded.bigWig
stromal.H3K4me1.signal_unstranded.bigWig
stromal.H3K4me3.signal_unstranded.bigWig
```

- https://epigenomesportal.ca/tracks/CEEHRC/hg38/
- Breast Basal CEMT0035
- Breast Stromal CEMT0036
- Breast Luminal CEMT0037
- Breast Luminal Progenitor CEMT0038 #### MCF10A Fastq

```
ls ~/CourseData/EPI_data/module123/fastq
MCF10A.ATAC.chr19.R1.fastq.gz
MCF10A.ATAC.chr19.R2.fastq.gz
MCF10A.H3K27ac.chr19.R1.fastq.gz
MCF10A.H3K27ac.chr19.R2.fastq.gz
MCF10A.H3K27me3.chr19.R1.fastq.gz
MCF10A.H3K27me3.chr19.R2.fastq.gz
MCF10A.H3K4me3.chr19.R1.fastq.gz
MCF10A.H3K4me3.chr19.R2.fastq.gz
MCF10A.Input.chr19.R1.fastq.gz
MCF10A.Input.chr19.R2.fastq.gz
```

- MCF10A histone marks and input come courtesy of Dr.Hirst
- ATACseq data originates from GSM6431322 #### Triplicates

```
CourseData/EPI_data/module123/triplicates/triplicates.csv

CourseData/EPI_data/module123/triplicates/alignments:
MCF10A_H3K4me3_chr19.CondA.Rep1.bam      MCF10A_H3K4me3_chr19.CondB.Rep2.bam      MCF1
MCF10A_H3K4me3_chr19.CondA.Rep1.bam.bai  MCF10A_H3K4me3_chr19.CondB.Rep2.bam.bai  MCF1
MCF10A_H3K4me3_chr19.CondA.Rep2.bam      MCF10A_H3K4me3_chr19.CondB.Rep3.bam      MCF1
MCF10A_H3K4me3_chr19.CondA.Rep2.bam.bai  MCF10A_H3K4me3_chr19.CondB.Rep3.bam.bai  MCF1
MCF10A_H3K4me3_chr19.CondA.Rep3.bam      MCF10A_input_chr19.CondA.Rep1.bam        MCF1
MCF10A_H3K4me3_chr19.CondA.Rep3.bam.bai  MCF10A_input_chr19.CondA.Rep1.bam.bai    MCF1
MCF10A_H3K4me3_chr19.CondB.Rep1.bam      MCF10A_input_chr19.CondA.Rep2.bam        MCF1
MCF10A_H3K4me3_chr19.CondB.Rep1.bam.bai  MCF10A_input_chr19.CondA.Rep2.bam.bai    MCF1

CourseData/EPI_data/module123/triplicates/bigWig:
CondA.Rep1.bw  CondA.Rep2.bw  CondA.Rep3.bw  CondB.Rep1.bw  CondB.Rep2.bw  CondB.Rep3.b

CourseData/EPI_data/module123/triplicates/peaks:
CondA.Rep1_peaks.narrowPeak  CondA.Rep3_peaks.narrowPeak  CondB.Rep2_peaks.narrowPeak
CondA.Rep2_peaks.narrowPeak  CondB.Rep1_peaks.narrowPeak  CondB.Rep3_peaks.narrowPeak
```

- triplicates were generated from MCF10A_H3K4me3 by choosing a list of exclusive peaks for condA and condB and subsampling replicates accordingly

Congratulations! You have completed Lab 1!

Chapter 5

Module 2: Alignment, Peak Calling, and Visualization

5.1 Lecture

5.2 Lab

5.2.1 Breakdown of Markdown File

- At the start of each step, the intention will declare.
- this is then follow by a code block

Code:

```
Like this! This is the main code to run for the step.

Additionally the code block will include a header to indicate what environment to the run code fo
###Shell###
pwd

###R###
getwd()
```

- explaining for commands will be broken down following code block
- `pwd` & `getwd()`
  - see your work directory
- sprinkled throughout will also include comments

Important points and considerations will also be raised as so.

5.2.2 Module 2. Peak Calling

5.2.2.1 Step 1: Dedup BAM file

- We first remove duplicates here b/c `MACS2`(peak caller) identifies duplicates via genomic coordinates (excessive removal of a lot reads).
- dedup now and instruct our peak caller to "keep duplicates" after.

Code:

```
###Shell###
treatment=MCF10A_H3K27ac_chr19
treatment_bam=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.bam
treatment_dedup=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.dedup.b
samtools view -@4 ${treatment_bam} -bh -q10 -F1028 -o ${treatment_dedup}

treatment=MCF10A_H3K27me3_chr19
treatment_bam=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.bam
treatment_dedup=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.dedup.b
samtools view -@4 ${treatment_bam} -bh -q10 -F1028 -o ${treatment_dedup}

treatment=MCF10A_H3K4me3_chr19
treatment_bam=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.bam
treatment_dedup=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.dedup.b
samtools view -@4 ${treatment_bam} -bh -q10 -F1028 -o ${treatment_dedup}

treatment=MCF10A_ATAC_chr19
treatment_bam=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.bam
treatment_dedup=~/workspace/module123/alignments/${treatment}.sorted.dup_marked.dedup.b
samtools view -@4 ${treatment_bam} -bh -q10 -F1028 -o ${treatment_dedup}

input=MCF10A_input_chr19
input_bam=~/workspace/module123/alignments/${input}.sorted.dup_marked.bam
input_dedup=~/workspace/module123/alignments/${input}.sorted.dup_marked.dedup.bam
samtools view -@4 ${input_bam} -bh -q10 -F1028 -o ${input_dedup}
```

- Run time <1 min per command
- Silently completes; no log needed
- `samtools view -@4 ${input_bam} -bh -q10 -F1028 -o ${input_dedup}`
  - we subset our file based on the following criteria
  - `-bh` we would like the output to be in `BAM` format and contain the header

  - `-q10` reads with mapping qual `>10`
  - `-F1028` any reads that do not have the flags `UNMAP` and `DUP`
    * whats the difference between `-f` and `-F`? ### Step 2A : Run Peak Caller (narrow)

- MACS has two modes for narrow marks and broad marks.
- Refer to this link to reference which mark are narrow or broad

Code:

```
###Shell###
mkdir -p ~/workspace/module123/peaks
name=MCF10A_H3K27ac
treatment=~/workspace/module123/alignments/${name}_chr19.sorted.dup_marked.dedup.bam
input=~/workspace/module123/alignments/MCF10A_input_chr19.sorted.dup_marked.dedup.bam


macs2 callpeak -t ${treatment} -c ${input} -f BAMPE -g 58617616 -n ${name} --keep-dup all --outdi

name=MCF10A_H3K4me3
treatment=~/workspace/module123/alignments/${name}_chr19.sorted.dup_marked.dedup.bam
input=~/workspace/module123/alignments/MCF10A_input_chr19.sorted.dup_marked.dedup.bam


macs2 callpeak -t ${treatment} -c ${input} -f BAMPE -g 58617616 -n ${name} --keep-dup all --outdi
```

- `macs2 callpeak -t ${treatment} -c ${input} -f BAMPE -g 58617616 -n ${name} --keep-dup all --outdir ~/workspace/module123/peaks/ --bdg 1> ~/workspace/module123/peaks/${name}.out.log 2> ~/workspace/module123/peaks/${name}.err.log`
  - `macs2 callpeak` General purpose peak calling mode
  - `-t ${treatment}` Treatment file, can provide more than one to "pool"
  - `-c ${input}` Control File/Input
  - `-f BAMPE` Instructs MACS2 on what kind of file to expect. Single/Paired-end bed/bam
  - `-g hs` Sets appropriate genome size for background sampling. Typically would set `hs=human` `mm=mouse` but in our case we use the HG38 size of chr19
  - `-n ${name}` name or prefix to use
  - `-q 0.05` FDR q value default
  - `--outdir ~/workspace/module123/peaks/` where to output files otherwise stores in current working directory
  - `--bdg` outputs pileup into bedgraph (a `BED` file where the fourth column is pileup/fragment count/coverage)

  - `1> ~/workspace/module123/peaks/${name}.out.log` output log
  - `2>  ~/workspace/module123/peaks/${name}.err.log` error log
  - let's inspect the peaks file
    * chromosome name
    * peak start
    * peak stop
    * peak name
    * int(-10*log10pvalue)
    * strand
    * Fold change at peak summit
    * -log10 P-value at Peak
    * -log10 Q-value at Peak
    * Summit position relative to peak

5.2.2.2 Step 2B : Run Peak Caller (broad)

Code:

```
###Shell###
mkdir -p ~/workspace/module123/peaks
name=MCF10A_H3K27me3
treatment=~/workspace/module123/alignments/${name}_chr19.sorted.dup_marked.dedup.bam
input=~/workspace/module123/alignments/MCF10A_input_chr19.sorted.dup_marked.dedup.bam

macs2 callpeak -t ${treatment} -c ${input} -f BAMPE -g 58617616 -n ${name} --keep-dup a
```

- `macs2 callpeak -t ${treatment} -c ${input} -f BAMPE -g 58617616 -n ${name} --keep-dup all --outdir ~/workspace/module123/peaks/ --bdg --broad 1> ~/workspace/module123/peaks/${name}.out.log 2> ~/workspace/module123/peaks/${name}.err.log`
  - `--broad` for broad marks - stitches small peaks together
  - let's note the differences between `broadPeak` vs `gappedPeak`
  - `gapped` has the additional columns starting from `strand` where most of the differences are visualization support for the narrrow peaks within the broadPeak
    * thickStart
    * thickEnd
    * itemRgb
    * blockCount
    * blockSizes
    * blockStarts
    * signalValue
    * pValue

    * qValue ### Step 3A : Run Peak Caller for ATAC - Make fragment file

- Convert our BAM to BED to easily manipulate coordinates
- perform shift to account for Tn5 binding as a dimer and inserts two adapters separated by 9 bp

This step can also be done for chipseq

Code:

```
name=MCF10A_ATAC
dedup=~/workspace/module123/alignments/${name}_chr19.sorted.dup_marked.dedup.bam
nsort=~/workspace/module123/alignments/${name}_chr19.nsorted.dup_marked.dedup.bam
tags=~/workspace/module123/peaks/${name}_chr19.frags.bed
samtools sort -@ 8 -n ${dedup} -o ${nsort}
bedtools bamtobed -bedpe -mate1 -i ${nsort} > ${tags}
```

- `samtools sort -@ 8 -n ${dedup} -o ${nsort}`
  - specifying the `-n` sorts according to read name rather than coordinates by default
- `bedtools bamtobed -bedpe -mate1 -i ${nsort} > ${tags}`
  - `bedtools bamtobed` converts our BAM to BED/coordinates for our reads
  - `-bedpe` instructs bedtools to report read pairs instead of each read individually
  - unpaired reads will emit a warning
  - `-mate1` instructs bedtools to report read1 information first followed by read2
  - columns : `chrR1,startR1,endR1,chrR2,startR2,endR2,readName,mapQ,strandR1,strandR2`
  - e.g. `chr19 4534039 4534190 chr19 4534110 4534248 SRR20814384.28 60 + -`
  - bedtools will check if file is name sorted or coordinate sorted ### Step 3B : Run Peak Caller for ATAC - Tn5 Shift
- perform shift to account for Tn5 binding as a dimer and inserts two adapters separated by 9 bp

Tn5 shift can be skipped if one is not interested in footprinting.

Code:

```
###Shell###
name=MCF10A_ATAC
```

```
tags=~/workspace/module123/peaks/${name}_chr19.frags.bed
tn5_tags=~/workspace/module123/peaks/${name}_chr19.frags.tn5.bed


cat ${tags} \
| awk 'BEGIN{{OFS="\t"}}{{printf "%s\t%s\t%s\tN\t1000\t%s\n%s\t%s\t%s\tN\t1000\t%s\n",$
| awk 'BEGIN{{OFS = "\t"}} {{if ($6 == "+") {{$2 = $2 + 4}} else if ($6 == "-") {{$3 =
> ${tn5_tags}
```

- `cat ${tags} | awk 'BEGIN{{OFS="\t"}}{{printf "%s\t%s\t%s\tN\t1000\t%s\n%s\t%s\t%s\t%s`
  `| awk 'BEGIN {{OFS = "\t"}} {{if ($6 == "+") {{$2 = $2 + 4}}`
  `else if ($6 == "-") {{$3 = $3 - 5}} if ($2 >= $3) {{ if ($6`
  `== "+") {{$2 = $3 - 1}} else {{$3 = $2 + 1}} }} print $0}}'`
  `> ${tn5_tags}`
  - `read tagFile | rearrange columns | shift Tag depending on strand`
  - see below for a more through breakdown of code

  ```
  awk '
  BEGIN{{OFS="\t"}}
  {
    {
        printf "%s\t%s\t%s\tN\t1000\t%s\n%s\t%s\t%s\tN\t1000\t%s\n",
        $1,$2,$3,$9,$4,$5,$6,$10
    }
  }'
  ```

- `BEGIN{{OFS="\t"}}` output file as tab delimited
- `printf "%s\t%s\t%s\tN\t1000\t%s\n%s\t%s\t%s\tN\t1000\t%s\n",$1,$2...`
  print string where `%s` is substituted with the next specified element
  - equivalent to `print chrR1,startR1,endR1,"N","1000",strandR1,chrR2,startR2,endR2,s`

  ```
  awk '
  BEGIN {{OFS = "\t"}}
  {
    {
        if ($6 == "+")
            {{$2 = $2 + 4}}
        else if ($6 == "-")
            {{$3 = $3 - 5}}
        if ($2 >= $3)
        {
            {
                if ($6 == "+")
                {{$2 = $3 - 1}}
  ```

```
                else
                {{$3 = $2 + 1}}
            }
        } print $0
    }
}'
> ${tn5_tags}
```

- if fragment is on `+` shifted 4 bp to the right
- if fragment is on `-` shift 5 bp to the left
  - while this is recommended, others have shifted `4/4` instead of `4/5`

- after the fix is start coordinates is greater than

5.2.2.3 Step 3C : Run Peak Caller for ATAC - Peak calling

Code:

```
###Shell###
name=MCF10A_ATAC
tn5_tags=~/workspace/module123/peaks/${name}_chr19.frags.tn5.bed

macs2 callpeak \
-t ${tn5_tags} \
-f BED \
-n ${name} \
-g 58617616 \
-p 0.01 \
--shift -100 \
--extsize 200 \
--nomodel \
--bdg \
--keep-dup all \
--outdir ~/workspace/module123/peaks/
```

- `macs2 callpeak -t /home/ubuntu/workspace/module123/peaks/MCF10A_ATAC_chr19.frags.tn5.bed -f BED -n {name} -g 58617616 -p 0.01 --nomodel --extsize 200 --shift -100 --bdg --keep-dup all`
  - `-f BED` we specify a `BED` format input instead of `BAM`
  - `-name ${name}` name of file
  - `-g 58617616` size of chr19
  - `-p 0.01` Pvalue cutoff
  - `--nomodel` off by default, normally calculates the `extsize` and `shift` parameters

  - `--extsize 200`, b/c Tn5's activity is on the `5'`, we extend the `5'` to get more representation
  - `--shift -100` should be `-1*(extsize/2)`, this focuses `MACS` on our `5'`
  - `--bdg` generate a pileup bedgraph
  - `--keep-dup all` retain "duplicates". As we've already filtered out duplicates, MACS2 will call duplicates via genomic coordinates
  - `--call-summits`

5.2.2.4 Step 4 : Blacklist removal

- Problematic regions can obscure our results, thus filter any peaks that coincide with those regions.

Code:

```
###Shell###
wget https://www.encodeproject.org/files/ENCFF356LFX/@@download/ENCFF356LFX.bed.gz -O

gunzip ~/workspace/module123/resources/hg38_blacklist.bed.gz

blacklist=~/workspace/module123/resources/hg38_blacklist.bed

sample="MCF10A_H3K27ac_peaks"
bedtools intersect -v -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklis
bedtools intersect -u -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklis

sample="MCF10A_H3K4me3_peaks"
bedtools intersect -v -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklis
bedtools intersect -u -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklis

sample="MCF10A_H3K27me3_peaks"
bedtools intersect -v -a ~/workspace/module123/peaks/${sample}.broadPeak -b ${blacklist
bedtools intersect -u -a ~/workspace/module123/peaks/${sample}.broadPeak -b ${blacklist

sample="MCF10A_ATAC_peaks"
bedtools intersect -v -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklis
bedtools intersect -u -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklis
```

- `bedtools intersect -u -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklist}`
  - `bedtools intersect` identify elements from `fileA` and `fileB` that overlap genomic coordiante wise

  - `-u` by default, bedtools will output every time a element intersection is detected i.e. if fileA_ele1 overlaps fileB_ele1 and fileB_ele2. The `-u` instead reports the element once regardless of how many overlaps

- `bedtools intersect -v -a ~/workspace/module123/peaks/${sample}.narrowPeak -b ${blacklist}`

  - `-v` reverse the behaviour, identify elements that do not overlap

- we'll return to this later when we can visualize the peaks

5.2.2.5 Step 5A : Visualization of pileup tracks

- in the next steps, we convert our pipleup bedgraphs and bed peak files into a smaller managable formats.

Code:

```
###Shell###
mkdir -p ~/workspace/module123/{bigBed,bigWig}
wget https://hgdownload.cse.ucsc.edu/goldenpath/hg38/bigZips/hg38.chrom.sizes -O workspace/module

sample="MCF10A_H3K27ac"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bedgraph=~/workspace/module123/peaks/${sample}_treat_pileup.bdg
output_bigwig=~/workspace/module123/bigWig/${sample}_treat_pileup.bw

sort -k1,1 -k2,2n ${input_bedgraph} > ~/workspace/module123/bigWig/tmp
bedGraphToBigWig ~/workspace/module123/bigWig/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigWig/tmp

input_bedgraph=~/workspace/module123/peaks/MCF10A_H3K27me3_control_lambda.bdg
output_bigwig=~/workspace/module123/bigWig/MCF10A_Input_control_pileup.bw

sort -k1,1 -k2,2n ${input_bedgraph} > ~/workspace/module123/bigWig/tmp
bedGraphToBigWig ~/workspace/module123/bigWig/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigWig/tmp
```

- `sort -k1,1 -k2,2n ${input_bedgraph}` sorting the BED file

  - `-k1,1` sort first by chromosome alphabetically
  - `-k2,2n` sort secondarily by genomic coordinates

- `bedGraphToBigWig` convert bedGraph file to bigWig

5.2.2.6 Step 5B : Visualization of pileup tracks continued

- we're converting the `bedgraph` file to a `bigWig` file.
- note `bedgraph` is an extension of bed(genomic coordinates) + a column with a numeric value
  - in our case pileup/coverage
- a bigWig is a binary version of a `wig` file
  - wig has a different format : https://useast.ensembl.org/info/website/upload/wig.html
  - the preferred convention for displaying data on a track

Code:

```
###Shell###
sample="MCF10A_H3K27me3"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bedgraph=~/workspace/module123/peaks/${sample}_treat_pileup.bdg
output_bigwig=~/workspace/module123/bigWig/${sample}_treat_pileup.bw

sort -k1,1 -k2,2n ${input_bedgraph} > ~/workspace/module123/bigWig/tmp
bedGraphToBigWig ~/workspace/module123/bigWig/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigWig/tmp

sample="MCF10A_H3K4me3"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bedgraph=~/workspace/module123/peaks/${sample}_treat_pileup.bdg
output_bigwig=~/workspace/module123/bigWig/${sample}_treat_pileup.bw

sort -k1,1 -k2,2n ${input_bedgraph} > ~/workspace/module123/bigWig/tmp
bedGraphToBigWig ~/workspace/module123/bigWig/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigWig/tmp

sample="MCF10A_ATAC"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bedgraph=~/workspace/module123/peaks/${sample}_treat_pileup.bdg
output_bigwig=~/workspace/module123/bigWig/${sample}_treat_pileup.bw

sort -k1,1 -k2,2n ${input_bedgraph} > ~/workspace/module123/bigWig/tmp
bedGraphToBigWig ~/workspace/module123/bigWig/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigWig/tmp
```

5.2.2.7 Step 5C : Visualization of peak tracks

- next we convert our `BED` files

Code:

```
###Shell###
mkdir -p ~/workspace/module123/{bigBed,bigWig}
wget https://hgdownload.cse.ucsc.edu/goldenpath/hg38/bigZips/hg38.chrom.sizes -O workspace/module

sample="MCF10A_H3K27ac"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bed=~/workspace/module123/peaks/${sample}_peaks.blacklistRemoved.narrowPeak
output_bigwig=~/workspace/module123/bigBed/${sample}.blacklistRemoved.bb


sort -k1,1 -k2,2n ${input_bed} | cut -f1-3 > ~/workspace/module123/bigBed/tmp
bedToBigBed ~/workspace/module123/bigBed/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigBed/tmp
```

- `sort -k1,1 -k2,2n ${input_bedgraph}` sorting the BED file
  - `-k1,1` sort first by chromosome alphabetically
  - `-k2,2n` sort secondarily by genomic coordinates
- `bedGraphToBigWig` convert bedGraph file to bigBed

5.2.2.8 Step 5D : Visualization of peak tracks continued

Code:

```
###Shell###
sample="MCF10A_ATAC"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bed=~/workspace/module123/peaks/${sample}_peaks.blacklistRemoved.narrowPeak
output_bigwig=~/workspace/module123/bigBed/${sample}.blacklistRemoved.bb


sort -k1,1 -k2,2n ${input_bed} | cut -f1-3 > ~/workspace/module123/bigBed/tmp
bedToBigBed ~/workspace/module123/bigBed/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigBed/tmp

sample="MCF10A_H3K4me3"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bed=~/workspace/module123/peaks/${sample}_peaks.blacklistRemoved.narrowPeak
output_bigwig=~/workspace/module123/bigBed/${sample}.blacklistRemoved.bb


sort -k1,1 -k2,2n ${input_bed} | cut -f1-3 > ~/workspace/module123/bigBed/tmp
```

```
bedToBigBed ~/workspace/module123/bigBed/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigBed/tmp

sample="MCF10A_H3K27me3"
chrom_sizes=~/workspace/module123/resources/hg38.chrom.sizes
input_bed=~/workspace/module123/peaks/${sample}_peaks.blacklistRemoved.broadPeak
output_bigwig=~/workspace/module123/bigBed/${sample}.blacklistRemoved.bb


sort -k1,1 -k2,2n ${input_bed} | cut -f1-3 > ~/workspace/module123/bigBed/tmp
bedToBigBed ~/workspace/module123/bigBed/tmp ${chrom_sizes} ${output_bigwig}
rm ~/workspace/module123/bigBed/tmp
```

5.2.2.9 Step 5E : Visualization of peaks and tracks

- can either download your tracks and load them from files or via URL
- colours used:
  - H3K27ac : blue
  - H3K27me3 : Brown
  - H3K4me3 : Green
  - ATAC : LightBlue
  - Black : Input

- BCL3 start site is enriched with H3K4me3 and H3K27ac and accessible

- Genes around RDH8 show enrichment of H3K27me3 and lack of accessibility and enrichment for H3K27ac and H3K4me3

- Blacklist identified region hightlighted in red, represented by both H3K27me3 and ATAC

5.2.2.10 Step 6A : Quality Control (Enrichment in key genomic areas)

- A way to determine the efficacy of your enrichment is to benchmark % of reads to called peaks (FRIP) or known regions
- in our toy example we'll be examining enrichment at promoters (TSS+/-2kb) and encode defining enhancer regions
- For H3K27me3 we'd typically look at HOX regions however no HOX regions on chr19

Code :

```
###Shell###
mkdir workspace/module123/qc
wget https://www.bcgsc.ca/downloads/esu/touchdown/hg38v79_genes_tss_2000.bed -O workspace/module1

sort -k1,1 -k2,2n workspace/module123/resources/hg38v79_genes_tss_2000.bed > tmp
mv tmp workspace/module123/resources/hg38v79_genes_tss_2000.bed

wget https://www.bcgsc.ca/downloads/esu/touchdown/encode_enhancers_liftover.bed -O workspace/modu

TSS=~/workspace/module123/resources/hg38v79_genes_tss_2000.bed
ENH=~/workspace/module123/resources/encode_enhancers_liftover.bed

sample="MCF10A_H3K27ac"
query_bam=~/workspace/module123/alignments/${sample}_chr19.sorted.dup_marked.bam

samtools view -@4 -q 10 -F 1028 $query_bam -c
samtools view -@4 -q 10 -F 1028 $query_bam -L ${ENH} -c
samtools view -@4 -q 10 -F 1028 $query_bam -L ~/workspace/module123/peaks/${sample}_peaks.blackli
```

- `samtools view -@4 -q 10 -F 1028 $query_bam -L ~/workspace/module123/peaks/${sample}_peaks.bl -c`
  - `samtools view` parse through our BAM file
  - `-@4` resources to use
  - `-q 10` filter for reads with mapping quality(MAPQ) >10
  - `-F 1028` Remove reads that have the following flags:UNMAP,DUP
  - `$query_bam` Bam of interest
  - `-c` count the number of reads that fulfil the criteria
  - `-L` perform actions on reads that fall within the specified genomic coordiantes (Bed File)

5.2.2.11 Step 6B : Quality Control (Enrichment in key genomic areas) Continued:

Code:

```
###Shell###
TSS=~/workspace/module123/resources/hg38v79_genes_tss_2000.bed
ENH=~/workspace/module123/resources/encode_enhancers_liftover.bed

for histone in H3K27ac H3K27me3 H3K4me3 ATAC input;
do
    sample="MCF10A_${histone}"
    query_bam=~/workspace/module123/alignments/${sample}_chr19.sorted.dup_marked.bam
    samtools view -@4 -q 10 -F 1028 $query_bam -c > ~/workspace/module123/qc/col_${histone}
```

```
    samtools view -@4 -q 10 -F 1028 $query_bam -L ${TSS} -c >> ~/workspace/module123/q
    samtools view -@4 -q 10 -F 1028 $query_bam -L ${ENH} -c >> ~/workspace/module123/q

    if [[ "$histone" == "H3K27me3" ]]; then
        peaks=~/workspace/module123/peaks/${sample}_peaks.blacklistRemoved.broadPeak
        samtools view -@4 -q 10 -F 1028 $query_bam -L ~/workspace/module123/peaks/${sa
    elif [[ "$histone" == "H3K27ac" || "$histone" == "H3K4me3" || "$histone" == "ATAC"
        peaks=~/workspace/module123/peaks/${sample}_peaks.blacklistRemoved.narrowPeak
        samtools view -@4 -q 10 -F 1028 $query_bam -L ~/workspace/module123/peaks/${sa
    else
        echo
    fi
done

paste ~/workspace/module123/qc/col_H3K27ac ~/workspace/module123/qc/col_H3K27me3 ~/wor
paste \
<(awk '{print $1/442829*100}' ~/workspace/module123/qc/col_H3K27ac) \
<(awk '{print $1/1610910*100}' ~/workspace/module123/qc/col_H3K27me3) \
<(awk '{print $1/1512352*100}' ~/workspace/module123/qc/col_H3K4me3) \
<(awk '{print $1/2751833*100}' ~/workspace/module123/qc/col_ATAC) \
<(awk '{print $1/1023265*100}' ~/workspace/module123/qc/col_input) > ~/workspace/modul

paste ~/workspace/module123/qc/col_H3K27ac ~/workspace/module123/qc/col_H3K27me3
~/workspace/module123/qc/col_H3K4me3 ~/workspace/module123/qc/col_ATAC
~/workspace/module123/qc/col_input > ~/workspace/module123/qc/integers.tsv
```

- `paste` lets us aggregate our results where each is a column) - - Reads enrichment in key region

	H3K27ac	H3K27me3	H3K4me3	ATAC	Input
Total	442829	1610910	1512352	2751833	1023265
TSS	127577	298326	1087032	924326	194996
Enhancer	242489	760089	834053	1721794	514115
In Peaks	69584	783294	1239717	1082674	

- Reads % enrichment in key region

	H3K27ac	H3K27me3	H3K4me3	ATAC	Input
TSS	28.8095	18.5191	71.8769	33.5895	19.0563
Enhancer	54.7591	47.1838	55.1494	62.569	50.2426
In Peaks	15.7135	48.6243	81.9728	39.3437	

- `H3K27ac` FRIP is poor as a result of the poorer sequencing depth.
- `H3K4me3` is performing well: high FRIP and enriched in promoter regions

5.2.3 Server resources

5.2.3.1 QC Resources

```
####TSS+/-2kb
mkdir workspace/module123/qc
wget https://www.bcgsc.ca/downloads/esu/touchdown/hg38v79_genes_tss_2000.bed -O workspace/module1

sort -k1,1 -k2,2n workspace/module123/resources/hg38v79_genes_tss_2000.bed > tmp
mv tmp workspace/module123/resources/hg38v79_genes_tss_2000.bed

####Enhancer liftover
wget https://www.bcgsc.ca/downloads/esu/touchdown/encode_enhancers_liftover.bed -O workspace/modu

####Blacklist
wget https://www.encodeproject.org/files/ENCFF356LFX/@@download/ENCFF356LFX.bed.gz -O ~/workspace

gunzip ~/workspace/module123/resources/hg38_blacklist.bed.gz
```

- `hg38v79_genes_tss_2000.bed`
  - Generated by downloading Ensemblv79 GTF convert to Bed +/-2kb of TSS. See https://www.biostars.org/p/56280/
- `encode_enhancers_liftover.bed`
  - download various ChroHMM state7 and merge

5.2.3.2 Encode Bed

```
ls ~/CourseData/EPI_data/module123/encode_bed
basal.H3K27ac.peak_calls.bed
basal.H3K27me3.peak_calls.bed
basal.H3K4me1.peak_calls.bed
basal.H3K4me3.peak_calls.bed
lp.H3K27ac.peak_calls.bed
lp.H3K27me3.peak_calls.bed
lp.H3K4me1.peak_calls.bed
lp.H3K4me3.peak_calls.bed
luminal.H3K27ac.peak_calls.bed
luminal.H3K27me3.peak_calls.bed
luminal.H3K4me1.peak_calls.bed
luminal.H3K4me3.peak_calls.bed
stromal.H3K27ac.peak_calls.bed
stromal.H3K27me3.peak_calls.bed
stromal.H3K4me1.peak_calls.bed
stromal.H3K4me3.peak_calls.bed
```

- https://epigenomesportal.ca/tracks/CEEHRC/hg38/
- Breast Basal CEMT0035
- Breast Stromal CEMT0036
- Breast Luminal CEMT0037
- Breast Luminal Progenitor CEMT0038 #### Encode BigWig

```
ls ~/CourseData/EPI_data/module123/encode_bigWig
basal.H3K27ac.signal_unstranded.bigWig
basal.H3K27me3.signal_unstranded.bigWig
basal.H3K4me1.signal_unstranded.bigWig
basal.H3K4me3.signal_unstranded.bigWig
lp.H3K27ac.signal_unstranded.bigWig
lp.H3K27me3.signal_unstranded.bigWig
lp.H3K4me1.signal_unstranded.bigWig
lp.H3K4me3.signal_unstranded.bigWig
luminal.H3K27ac.signal_unstranded.bigWig
luminal.H3K27me3.signal_unstranded.bigWig
luminal.H3K4me1.signal_unstranded.bigWig
luminal.H3K4me3.signal_unstranded.bigWig
stromal.H3K27ac.signal_unstranded.bigWig
stromal.H3K27me3.signal_unstranded.bigWig
stromal.H3K4me1.signal_unstranded.bigWig
stromal.H3K4me3.signal_unstranded.bigWig
```

- https://epigenomesportal.ca/tracks/CEEHRC/hg38/
- Breast Basal CEMT0035
- Breast Stromal CEMT0036
- Breast Luminal CEMT0037
- Breast Luminal Progenitor CEMT0038 #### MCF10A Fastq

```
ls ~/CourseData/EPI_data/module123/fastq
MCF10A.ATAC.chr19.R1.fastq.gz
MCF10A.ATAC.chr19.R2.fastq.gz
MCF10A.H3K27ac.chr19.R1.fastq.gz
MCF10A.H3K27ac.chr19.R2.fastq.gz
MCF10A.H3K27me3.chr19.R1.fastq.gz
MCF10A.H3K27me3.chr19.R2.fastq.gz
MCF10A.H3K4me3.chr19.R1.fastq.gz
MCF10A.H3K4me3.chr19.R2.fastq.gz
MCF10A.Input.chr19.R1.fastq.gz
MCF10A.Input.chr19.R2.fastq.gz
```

- MCF10A histone marks and input come courtesy of Dr.Hirst
- ATACseq data originates from GSM6431322 #### Triplicates

```
CourseData/EPI_data/module123/triplicates/triplicates.csv

CourseData/EPI_data/module123/triplicates/alignments:
MCF10A_H3K4me3_chr19.CondA.Rep1.bam      MCF10A_H3K4me3_chr19.CondB.Rep2.bam      MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep1.bam.bai  MCF10A_H3K4me3_chr19.CondB.Rep2.bam.bai  MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep2.bam      MCF10A_H3K4me3_chr19.CondB.Rep3.bam      MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep2.bam.bai  MCF10A_H3K4me3_chr19.CondB.Rep3.bam.bai  MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep3.bam      MCF10A_input_chr19.CondA.Rep1.bam        MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep3.bam.bai  MCF10A_input_chr19.CondA.Rep1.bam.bai    MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondB.Rep1.bam      MCF10A_input_chr19.CondA.Rep2.bam        MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondB.Rep1.bam.bai  MCF10A_input_chr19.CondA.Rep2.bam.bai    MCF10A_input_ch

CourseData/EPI_data/module123/triplicates/bigWig:
CondA.Rep1.bw  CondA.Rep2.bw  CondA.Rep3.bw  CondB.Rep1.bw  CondB.Rep2.bw  CondB.Rep3.bw

CourseData/EPI_data/module123/triplicates/peaks:
CondA.Rep1_peaks.narrowPeak  CondA.Rep3_peaks.narrowPeak  CondB.Rep2_peaks.narrowPeak
CondA.Rep2_peaks.narrowPeak  CondB.Rep1_peaks.narrowPeak  CondB.Rep3_peaks.narrowPeak
```

- triplicates were generated from MCF10A_H3K4me3 by choosing a list of exclusive peaks for condA and condB and randomly subsampling replicates accordingly

Congratulations! You have completed Lab 2!

Chapter 6

Module 3: ChIP-seq Differential Analysis

6.1 Lecture

6.2 Lab

6.2.1 Breakdown of markdown file

- At the start of each step, the intention will declare.
- this is then follow by a code block

Code:

```
Like this! This is the main code to run for the step.

Additionally the code block will include a header to indicate what environment to the run code fo
###Shell###
pwd

###R###
getwd()
```

- explaining for commands will be broken down following code block
- `pwd` & `getwd()`
  - see your work directory
- sprinkled throughout will also include comments

Important points and considerations will also be raised as so.

6.2.2 Module 3 - Differential Analysis

6.2.2.1 Step1A: Copy bigWig resources

- we'll load some bigWigs to compare with

Code

```
###Shell###
cp ~/CourseData/EPI_data/module123/encode_bigWig/*H3K4me3* ~/workspace/module123/bigWig
cp ~/CourseData/EPI_data/module123/encode_bigBed/*H3K4me3* ~/workspace/module123/bigBed
cp ~/CourseData/EPI_data/module123/triplicates/bigWig/* ~/workspace/module123/bigWig/
```

6.2.2.2 Step1B: Using Bedtools to compare marks

- We'll explore how to use bedtools to compare MCF10A histone marks and the interpretation of results

Code

```
###Shell###
MCF10A_H3K27ac=~/workspace/module123/peaks/MCF10A_H3K27ac_peaks.blacklistRemoved.narrow
MCF10A_H3K27me3=~/workspace/module123/peaks/MCF10A_H3K27me3_peaks.blacklistRemoved.broa
MCF10A_H3K4me3=~/workspace/module123/peaks/MCF10A_H3K4me3_peaks.blacklistRemoved.narrow

bedtools intersect -u -a ${MCF10A_H3K27ac} -b ${MCF10A_H3K27me3} | wc -l
bedtools intersect -u -a ${MCF10A_H3K27ac} -b ${MCF10A_H3K4me3} | wc -l
```

- `bedtools intersect -u -a ${MCF10A_H3K27ac} -b ${MCF10A_H3K27me3} | wc -l`
  - results in an intersect of `6/988`
- `bedtools intersect -u -a ${MCF10A_H3K27ac} -b ${MCF10A_H3K4me3} | wc -l`
  - results in an intersect of `789/988`
- what do we know about the relationship of H3K27ac vs H3K27me3 vs H3K4me3?
  - H3K27ac and H3K27me3 tend to be antagonistic, hence the very small intersect
  - H3K27ac co-occurs with H3K4me3 at promoters, hence the larger intersect

6.2.2.3 Step1C: Using Bedtools to compare samples

- We'll demonstrate how to do comparisons on mass against ENCODE breast data and interpret results

Code

```
###Shell###
basal_H3K27ac=~/CourseData/EPI_data/module123/encode_bed/basal.H3K27ac.peak_calls.bed
luminal_H3K27ac=~/CourseData/EPI_data/module123/encode_bed/luminal.H3K27ac.peak_calls.bed
stromal_H3K27ac=~/CourseData/EPI_data/module123/encode_bed/stromal.H3K27ac.peak_calls.bed
lp_H3K27ac=~/CourseData/EPI_data/module123/encode_bed/lp.H3K27ac.peak_calls.bed
MCF10A_H3K27ac=~/workspace/module123/peaks/MCF10A_H3K27ac_peaks.blacklistRemoved.narrowPeak

paste \
<(ls ~/CourseData/EPI_data/module123/encode_bed/*H3K4me3* | xargs -I {} sh -c "bedtools intersect
<(ls ~/CourseData/EPI_data/module123/encode_bed/*H3K27ac* |  xargs -I {} sh -c "bedtools intersec
<(ls ~/CourseData/EPI_data/module123/encode_bed/*H3K27me3* |  xargs -I {} sh -c "bedtools interse
```

- `paste \ <(ls CourseData/module123/bed/*H3K4me3* | xargs -I {} sh -c "bedtools intersect -u -a workspace/module123/peaks/MCF10A_H3K4me3_peaks.blacklistR -b {} | wc -l") \ <(ls CourseData/module123/bed/*H3K27ac* | xargs -I {} sh -c "bedtools intersect -u -a workspace/module123/peaks/MCF10A_H3K27ac_peaks.b -b {} | wc -l")`
  - `paste` lets us aggregate our results where each column is the output from the command within `<(COMMAND)`
  - Each `<(COMMAND)`, contains the following `LOOKUP | FORLOOP intersect and count`
  - `\` let's continue our command on another line
- Intersect numbers:

	H3K4me3	H3K27ac	H3K27me3
MCF10A	1406	988	4797
Intersecting Basal	1039	656	2420
Intersecting Luminal Progenitor	1099	778	2496
Intersecting Luminal	1024	766	2430
Intersecting Stromal	978	717	2604

- MCF10A is luminal progenitor like, how is that relationship reflect in the epigenetic landscape?
  - higher amount of intersect in permissive marks of H3K4me3 and H3K27ac

6.2.2.4 Step1D: Using Bedtools and pipe

- We'll demonstrate advanced queries by piping and doing multiple bedtool queries

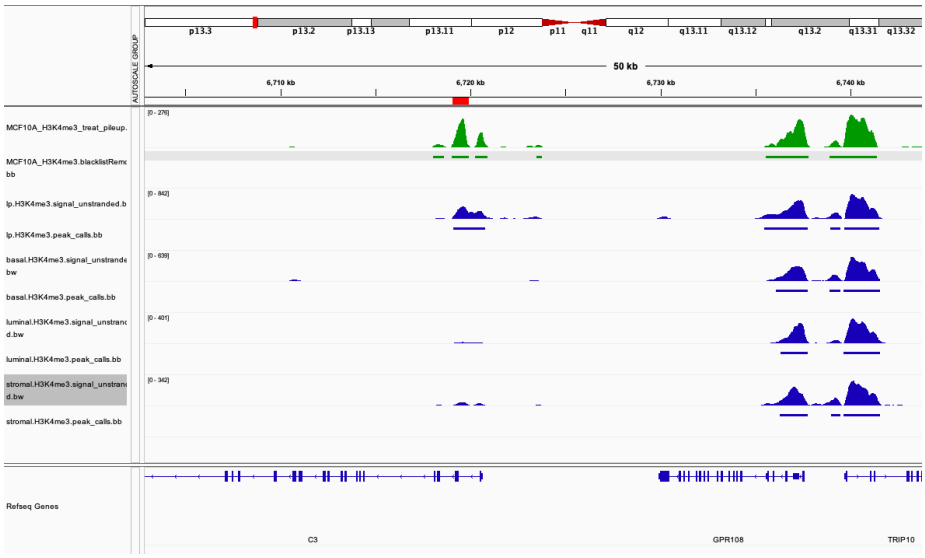
Code:

```
###Shell###
basal_H3K4me3=~/CourseData/EPI_data/module123/encode_bed/basal.H3K4me3.peak_calls.bed
luminal_H3K4me3=~/CourseData/EPI_data/module123/encode_bed/luminal.H3K4me3.peak_calls.
stromal_H3K4me3=~/CourseData/EPI_data/module123/encode_bed/stromal.H3K4me3.peak_calls.
lp_H3K4me3=~/CourseData/EPI_data/module123/encode_bed/lp.H3K4me3.peak_calls.bed
MCF10A_H3K4me3=~/workspace/module123/peaks/MCF10A_H3K4me3_peaks.blacklistRemoved.narrow

bedtools intersect -u -a ${MCF10A_H3K4me3} -b ${lp_H3K4me3} | bedtools intersect -v -a

bedtools intersect -u -a ${MCF10A_H3K4me3} -b ${lp_H3K4me3} |\
bedtools intersect -u -a stdin -b ${basal_H3K4me3} |\
bedtools intersect -u -a stdin -b ${luminal_H3K4me3} |\
bedtools intersect -u -a stdin -b ${stromal_H3K4me3} | wc -l
```

- `bedtools intersect -u -a ${MCF10A_H3K4me3} -b ${lp_H3K4me3} | bedtools intersect -v -a stdin -b ${basal_H3K4me3} ${luminal_H3K4me3} ${stromal_H3K4me3}| wc -l`
  - The command can be broken down into the following : `Common with LP | Not found in Basal OR Luminal OR stromal`
  - `stdin` takes the results from half of our command and utilizies as input in the next
  - Other tools may have a similar function of `stdin`. Check documentation first.



- `bedtools intersect -u -a ${MCF10A_H3K4me3} -b ${lp_H3K4me3} | bedtools intersect -u -a stdin -b {basal_H3K4me3} | bedtools intersect -u -a stdin -b ${luminal_H3K4me3} | bedtools intersect -u -a stdin -b ${stromal_H3K4me3} | wc -l`

  - The command can be broken down into the following : `MCF10A Common with LP | Common with Basal | Common with Luminal | Stromal`

6.2.2.5 Step1E: Using Bedtools to compare binary conditions/models

- We'll explore how to use bedtools to compare binary conditions and possible interpretations

Code:

```
###Shell###
condA_rep1=~/CourseData/EPI_data/module123/triplicates/peaks/CondA.Rep1_peaks.narrowPea
condB_rep1=~/CourseData/EPI_data/module123/triplicates/peaks/CondB.Rep1_peaks.narrowPea
condA_rep2=~/CourseData/EPI_data/module123/triplicates/peaks/CondA.Rep2_peaks.narrowPea
condB_rep2=~/CourseData/EPI_data/module123/triplicates/peaks/CondB.Rep2_peaks.narrowPea
condA_rep3=~/CourseData/EPI_data/module123/triplicates/peaks/CondA.Rep3_peaks.narrowPea
condB_rep3=~/CourseData/EPI_data/module123/triplicates/peaks/CondB.Rep3_peaks.narrowPea

bedtools intersect -u -a ${condA_rep1} -b ${condA_rep2} | wc -l
bedtools intersect -u -a ${condA_rep1} -b ${condB_rep2} | wc -l
```

```
bedtools intersect -v -a ${condA_rep1} -b ${condA_rep2} | wc -l
bedtools intersect -v -a ${condA_rep1} -b ${condB_rep2} | wc -l

bedtools intersect -u -a ${condA_rep1} -b ${condA_rep2} ${condA_rep3} | wc -l

bedtools intersect -u -a ${condA_rep1} -b ${condA_rep2} ${condA_rep3} -f 0.5 -F 0.5 | wc -l

bedtools intersect -wao -a ${condA_rep1} -b ${condA_rep2} ${condA_rep3} | head
```

- `bedtools intersect -u -a ${condA_rep1} -b ${condA_rep2} | wc -l`
  - counting the number of condA_rep1 peaks that intersect condA_rep2
  - returns `1191`
- `bedtools intersect -u -a ${condA_rep1} -b ${condB_rep2} | wc -l`
  - counting the number of condA_rep1 peaks that intersect condB_rep2
  - returns `1093`
- `bedtools intersect -v -a ${condA_rep1} -b ${condA_rep2} | wc -l`
  - counting the number of condA_rep1 peaks that do not intersect condA_rep2
  - return `50`
- `bedtools intersect -v -a ${condA_rep1} -b ${condB_rep2} | wc -l`
  - counting the number of condA_rep1 peaks that do not intersect condB_rep2
  - returns `148`
- as expected our replicates of matching conditions have more in common
- `bedtools intersect -u -a ${condA_rep1} -b ${condA_rep2} ${condA_rep3} | wc -l`
  - counting the number of condA_rep1 peaks that intersect condA_rep2 or condA_rep3
- `bedtools intersect -wao -a ${condA_rep1} -b ${condA_rep2} ${condA_rep3} | head`
  - specify `-wao` returns the original line of `${condA_rep1}` and the element it intersects
  - additionally adds an identify column for the database and number of base pairs overlapping

- `bedtools intersect -u -a ${condA_rep1} -b ${condA_rep2} ${condA_rep3} -f 0.5 -F 0.5 | wc -l`
  - the flag `-f 0.5` adds the conditions that inte##rsects are only counted when 50% overlap of A occurs
  - the flag `-F 0.5` adds the conditions that intersects are only counted when 50% overlap of B occurs
  - if we remove one of the flags, how does the number change?
  - what if we wanted integer threshold instead of percentage?
- `bedtools intersect -wao -a ${condA_rep1} -b ${condA_rep2} ${condA_rep3} | head`
  - specify `-wao` returns the original line of `${condA_rep1}` and the element it intersects
  - additionally adds an identify column for the database and number of base pairs overlapping

6.2.2.6 Step1F: Other useful bedtool functions

- We'll highlight other useful bedtool applications.

Code:

```
###Shell###
MCF10A_H3K27ac=~/workspace/module123/peaks/MCF10A_H3K27ac_peaks.blacklistRemoved.narrow
TSS=~/workspace/module123/resources/hg38v79_genes_tss_2000.bed

bedtools closest -a ${MCF10A_H3K27ac} -b ${TSS} -d | head

###
condA_peaks=~/CourseData/EPI_data/module123/triplicates/peaks/CondA.Rep1_peaks.narrowPe
condB_peaks=~/CourseData/EPI_data/module123/triplicates/peaks/CondB.Rep1_peaks.narrowPe

cat ${condA_peaks} ${condB_peaks} | sort -k1,1 -k2,2n | bedtools merge -i stdin > ~/wor

####
methylation=~/workspace/module123/resources/example_methylation.bed

echo chr19 1 58617616 |\
sed 's/ /\t/g' |\
bedtools makewindows -w 50 -b stdin |\
awk 'BEGIN{{srand(1)}}{print $0"\t"rand()}' \
> ${methylation}

bedtools map -a ${MCF10A_H3K27ac} -b ${methylation} -c 4 -o median,count | head
```

- `sed 's/ /\t/g'` replace a with `\t`
- `bedtools closest -a ${MCF10A_H3K27ac} -b ${TSS} -d | head`
  - Identifies features in fileB that are closes to fileA
  - useful for mapping enhancers to their closest transcription start site
  - `-d` will make the tool report the distance
  - in our example, if the distance is zero the H3K27ac instead reflects an activated promoter.
- `cat ${condA_rep1} ${condA_rep2} ${condA_rep3} | sort -k1,1 -k2,2n | bedtools merge -i stdin | wc -l`
  - Pseudo code : `read peak files | sort peak files | merge peak files | line count`
  - `bedtools merge` takes elements specified in the input and merges the features if they intersect
  - behaviour can be modified to require a certain mount overlap or bookended features
  - compare how many peaks `cat ${condA_rep1} ${condA_rep2} ${condA_rep3}` starts off with
  - following merge how many peaks are left?
- `echo chr19\\t1\\t58617616 | bedtools makewindows -w 50 -b stdin | awk 'BEGIN{srand(1);}{print $0"\t"rand()}' > ${metylation}`
  - Pseudo code : `simulate a bed file of chr19 | turn bedFile into 50bp bins | add random float value`
  - `echo chr19\\t1\\t58617616`
    - '`\\t` an escape character is uses to generate a tab
  - `bedtools makewindows -w 50 -b stdin`
    - `-w 50` specifies our window size
    - can alternatively use `-n` to specified how many windows we want to divide our bedFile into
  - `awk 'BEGIN{srand(1);}{print $0"\t"rand()}'`
    - read our input (the 50bp window bed file of chr19) and generate a random float
    - `srand(1)` set our seed number. Changing this will affect our pseudo random numbers
    - `print $0"\t"rand()` as `awk` process line by line, print the current line (the windowed genomic coordiantes) and a random float number
  - `bedtools map -a ${MCF10A_H3K27ac} -b ${methylation} -c 4 -o median,count`
    - `bedtools map` apply a function summarizing the values of fileB that intersect fileA

    - in our example for we're looking at the methylation of H3K27ac peaks
    - `-c 4` indicates which columns from fileB we'd like to use
    - `-o median,count` return the median of col 4 and coutn the number of elements from fileB that intersected the particular element from fileA

6.2.2.7 Step2: Differential peaks utilizing triplicates and DiffBind

- We'll perform analysis on mock MFC10A H3K4me3 data to get significant differential peaks for each condition. To do so, we'll utilizing the `diffBind` package in R

Code :

```
###R###
library(DiffBind)
setwd("/home/ubuntu")
samples <- read.csv("CourseData/EPI_data/module123/triplicates/triplicates.csv")
MCF10A <- dba(sampleSheet=samples)
MCF10A <- dba.count(MCF10A, bUseSummarizeOverlaps=TRUE)
dba.plotPCA(MCF10A, attributes=DBA_CONDITION,label=DBA_ID)
plot(MCF10A)
MCF10A <- dba.contrast(MCF10A, categories=DBA_CONDITION)

MCF10A <- dba.analyze(MCF10A, method=DBA_EDGER)

analyzed_peaks <- dba.report(MCF10A, method=DBA_EDGER, fold=1)

dba.plotMA(MCF10A, bXY=TRUE , method=DBA_EDGER, fold=1)

write.table(analyzed_peaks, file="workspace/module123/diffBind/differential_peaks.tsv"
```

- `library(DiffBind)`
    - we load R package DiffBind
- `setwd("/home/ubuntu")`
    - set our working directory
- `read.csv("CourseData/EPI_data/module123/triplicates/triplicates.csv")`
    - read in our csv
    - let's inspect the columns
- `MCF10A <- dba(sampleSheet=samples)`

  - read our samplesheet into the `dba` object that will be saved as `MCF10A`

- `MCF10A <- dba.count(MCF10A, bUseSummarizeOverlaps=TRUE)`
  - count the number of fragments that intersect with peaks
  - `bUseSummarizeOverlaps=TRUE` indicates the counting module to be used. `SummarizeOverlaps` comes from GenomicAlignments.

- `dba.plotPCA(MCF10A, attributes=DBA_CONDITION,label=DBA_ID)`
  - generate a principle component analysis using data from our object `MCF10A` where the annotations are `DBA_CONDITION` and the labelling is `DBA_ID`

- `plot(MCF10A)`
  - generate a heatmap with correlation and dendrogram
  - should note the correlation is based on score of overlap and not pearson and spearman, should recalculate

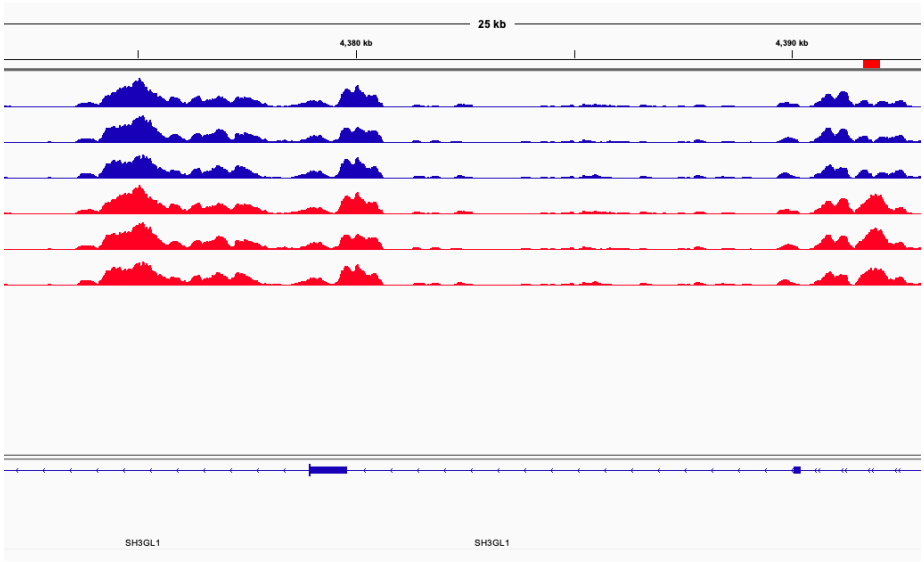
- `MCF10A <- dba.contrast(MCF10A, categories=DBA_CONDITION)`
  - declares what are the conditions for our differential groups
  - `categories=DBA_CONDITION` the category we want to compare

- `MCF10A <- dba.analyze(MCF10A, method=DBA_EDGER)`
  - perform an analysis based on the `contrast` we previously established.
  - `method=DBA_EDGER`, analysis engine is a library called `edgeR`
  - note for our specific example `deseq2` does not work. `Deseq2` has a built in check for variablity which our synthetic dataset is lacking

- `analyzed_peaks <- dba.report(MCF10A, method=DBA_EDGER, fold=1)`
  - report the peaks identified by `DBA_EDGER` to be significant and have an absolute fold change `>1`

- `dba.plotMA(MCF10A, bXY=TRUE , method=DBA_EDGER, fold=1)`
  - generates Scatter plot
  - `method=DBA_EDGER` fetch results based on our previous analysis using `edgeR`
  - `bXY=TRUE` produces a scatter plot, `FALSE` produces a MA plot
  - `fold=1` report differential positions that meet fold change threshold

- `write.table(analyzed_peaks, file="workspace/module123/diffBind/differential_peaks.tsv", sep="\t", quote=F, row.names=F, col.names=T)`
  - save our differential peaks to a TSV
  - `sep="\t"` the seperator to be used
  - `col.names=T` include column names
  - `row.names=F` include row names
  - `quote=F` if we want to include quotations around values



6.2.2.8 Step3A: Differential peaks utilizing Fold change and significance - Merged peaks

- Previously we performed differential analysis on triplicates, now let's explore how to do so on two samples.
- We'll combine our two peak sets into an unified set.

Code:

```
###Shell###
condA_peaks=~/CourseData/EPI_data/module123/triplicates/peaks/CondA.Rep1_peaks.narrowPe
condB_peaks=~/CourseData/EPI_data/module123/triplicates/peaks/CondB.Rep1_peaks.narrowPe

cat ${condA_peaks} ${condB_peaks} | sort -k1,1 -k2,2n | bedtools merge -i stdin > ~/wor
```

- `cat ${condA_peaks} ${condB_peaks} | sort -k1,1 -k2,2n | bedtools merge -i stdin > merged_peaks.bed`
  - Pseudo code break down : `Read our peaks | sort peaks coordinate wise | merge peaks`

6.2.2.9 Step3B: Differential peaks utilizing Fold change and significance - Merged peaks

- We'll combine our two peak sets into an unified set.

Code:

```
###Shell###
condA_peaks=~/CourseData/EPI_data/module123/triplicates/peaks/CondA.Rep1_peaks.narrowPeak
condB_peaks=~/CourseData/EPI_data/module123/triplicates/peaks/CondB.Rep1_peaks.narrowPeak
mkdir ~/workspace/module123/edgeR/
cat ${condA_peaks} ${condB_peaks} | sort -k1,1 -k2,2n | bedtools merge -i stdin > ~/workspace/mod
```

- `cat ${condA_peaks} ${condB_peaks} | sort -k1,1 -k2,2n | bedtools merge -i stdin > merged_peaks.bed`
  - Pseudo code break down : `Read our peaks | sort peaks coordinate wise | merge peaks`

6.2.2.10 Step3B: Differential peaks utilizing Fold change and significance - read counts per peak

- We'll derrive RPKM values per BAM for each peak in our peak set

Code:

```
###Shell###
peaks=~/workspace/module123/edgeR/merged_peaks.bed
condA_bam=~/workspace/module123/alignments/MCF10A_H3K4me3_chr19.CondA.Rep1.bam
condB_bam=~/workspace/module123/alignments/MCF10A_H3K4me3_chr19.CondB.Rep1.bam

condA_count=~/workspace/module123/edgeR/MCF10A_H3K4me3_chr19.CondA.Rep1.bed
condB_count=~/workspace/module123/edgeR/MCF10A_H3K4me3_chr19.CondB.Rep1.bed

bedtools intersect -a ${peaks} -b ${condA_bam} -c > ${condA_count}
bedtools intersect -a ${peaks} -b ${condB_bam} -c > ${condB_count}
```

- `bedtools intersect -a ${peaks} -b ${condA_bam} -c > ${condA_count}`
  - our intersect command is the same in principle but we're intersecting our peaks with a `BAM` file
  - `-c` reports counts of our `BAM` that overlap our peaks

6.2.2.11 Step3C: Differential peaks utilizing Fold change and significance - EdgeR differential

- we'll read our data into `R` and perform a statistical analysis

Code:

###R### setwd("/home/ubuntu") library(edgeR) library(dplyr)

condA<-read.csv("workspace/module123/edgeR/MCF10A_H3K4me3_chr19.CondA.Rep1.bed",sep='', = c("chr", "start", "end","MCF10A_H3K4me3_chr19.CondA.Rep1"),colClasses= c("character","character","character","numeric")) condB<-read.csv("workspace/module123/edgeR/MC = c("chr", "start", "end","MCF10A_H3K4me3_chr19.CondB.Rep1"),colClasses= c("character","character","character","numeric"))

peaks<-data.frame(MCF10A_H3K4me3_chr19.CondA.Rep1=condA$MCF10A_H3K4me3_chr19.CondA condB$MCF10A_H3K4me3_chr19.CondB.Rep1) row.names(peaks)<- paste(condA$chr, condA$start,condA$end,sep='_')

edger_dl <- DGEList(counts=peaks, group=1:2,lib.size=c(1131503,1266436))

edger_tmm <- calcNormFactors(edger_dl, method = "TMM")

bvc=0.1

edger_et <- exactTest(edger_tmm,dispersion=bvc^2)

edger_tp <- topTags(edger_et, n=nrow(edger_et$table),adjust.method="BH")

de <- edger_tp$table %>% filter(FDR < 0.01) %>% filter(logFC >=1 | logFC <=-1)

write.table(de, file="workspace/module123/edgeR/differential_peaks_edger.tsv", sep="", quote=F, row.names=F, col.names=F)

- `condA<-read.csv("workspace/module123/edgeR/MCF10A_H3K4me3_chr19.CondA.Rep1.bed",sep=
    - `read.csv` read the `CSV` file into a `data.frame`
    - `sep='\t'` specify the delimiter
    - `col.names = c("chr", "start", "end","MCF10A_H3K4me3_chr19.CondA.Rep1")` set our
    - `colClasses= c("character","character","character","numeric")` indicate with colu
- `peaks<-data.frame(MCF10A_H3K4me3_chr19.CondA.Rep1=condA$MCF10A_H3K4me3_chr19.CondA.
    - make a new data.frame using our two columns from condA and condB
- `row.names(peaks)<-paste(peaks$chr,peaks$start,peaks$end,sep='_')`
    - edit row names to be match genomic coordinates
    - ``row.names(peaks)<-` indicate we want to overwrite exist row names
    - `paste(peaks$chr,peaks$start,peaks$end,sep='_')` we want to concatenate our `chr
- `edger_dl <- DGEList(counts=peaks, group=1:2,lib.size=c(1131503,1266436))`
    - `DGEList(counts=peaks_clean, group=1:2)` read in our `peak_clean` data.frame
    - `group=1:2` identify which groups we want to contrast
    - `lib.size=c(1131503,1266436)` library size for the two conditions
- `edger_tmm <- calcNormFactors(edger_dl, method = "TMM")` calculate the normalization
- `bvc=0.01` Set the `square-rootdispersion`. according to edgeR documentation: `from
- `edger_et <- exactTest(edger_dl,dispersion=bcv^2)`
    - calculate `FC` and `Pvalue` per row
- `edger_tp <- topTags(edger_et, n=nrow(edger_et$table),adjust.method="BH")`

```
    - calcualte `FDR` via Benjamini-Hochberg
- `de <- edger_tp$table %>% filter(FDR < 0.01) %>% filter(logFC >=1 | logFC <=-1)`
    - `filter(FDR < 0.01)` filter for significant peaks
    - `filter(logFC >=1 | logFC <=-1)` filter for peaks with appropriate fold change
- `write.table(de, file="workspace/module123/edgeR/differential_peaks_edger.tsv", sep="\t", quote
    - save files
<img src="https://github.com/bioinformaticsdotca/EPI_2023/blob/module123/module123_images/edger.p
```

6.2.3 Server resources

6.2.3.1 QC Resources

```
###TSS+/-2kb
mkdir workspace/module123/qc
wget https://www.bcgsc.ca/downloads/esu/touchdown/hg38v79_genes_tss_2000.bed -O workspace/module1

sort -k1,1 -k2,2n workspace/module123/resources/hg38v79_genes_tss_2000.bed > tmp
mv tmp workspace/module123/resources/hg38v79_genes_tss_2000.bed

###Enhancer liftover
wget https://www.bcgsc.ca/downloads/esu/touchdown/encode_enhancers_liftover.bed -O workspace/modu

###Blacklist
wget https://www.encodeproject.org/files/ENCFF356LFX/@@download/ENCFF356LFX.bed.gz -O ~/workspace

gunzip ~/workspace/module123/resources/hg38_blacklist.bed.gz
```

- `hg38v79_genes_tss_2000.bed`
  - Generated by downloading Ensemblv79 GTF convert to Bed +/-2kb of TSS. See https://www.biostars.org/p/56280/
- `encode_enhancers_liftover.bed`
  - download various ChroHMM state7 and merge

6.2.3.2 Encode Bed

```
ls ~/CourseData/EPI_data/module123/encode_bed
basal.H3K27ac.peak_calls.bed
basal.H3K27me3.peak_calls.bed
basal.H3K4me1.peak_calls.bed
basal.H3K4me3.peak_calls.bed
lp.H3K27ac.peak_calls.bed
lp.H3K27me3.peak_calls.bed
```

```
lp.H3K4me1.peak_calls.bed
lp.H3K4me3.peak_calls.bed
luminal.H3K27ac.peak_calls.bed
luminal.H3K27me3.peak_calls.bed
luminal.H3K4me1.peak_calls.bed
luminal.H3K4me3.peak_calls.bed
stromal.H3K27ac.peak_calls.bed
stromal.H3K27me3.peak_calls.bed
stromal.H3K4me1.peak_calls.bed
stromal.H3K4me3.peak_calls.bed
```

- https://epigenomesportal.ca/tracks/CEEHRC/hg38/
- Breast Basal CEMT0035
- Breast Stromal CEMT0036
- Breast Luminal CEMT0037
- Breast Luminal Progenitor CEMT0038 #### Encode BigWig

```
ls ~/CourseData/EPI_data/module123/encode_bigWig
basal.H3K27ac.signal_unstranded.bigWig
basal.H3K27me3.signal_unstranded.bigWig
basal.H3K4me1.signal_unstranded.bigWig
basal.H3K4me3.signal_unstranded.bigWig
lp.H3K27ac.signal_unstranded.bigWig
lp.H3K27me3.signal_unstranded.bigWig
lp.H3K4me1.signal_unstranded.bigWig
lp.H3K4me3.signal_unstranded.bigWig
luminal.H3K27ac.signal_unstranded.bigWig
luminal.H3K27me3.signal_unstranded.bigWig
luminal.H3K4me1.signal_unstranded.bigWig
luminal.H3K4me3.signal_unstranded.bigWig
stromal.H3K27ac.signal_unstranded.bigWig
stromal.H3K27me3.signal_unstranded.bigWig
stromal.H3K4me1.signal_unstranded.bigWig
stromal.H3K4me3.signal_unstranded.bigWig
```

- https://epigenomesportal.ca/tracks/CEEHRC/hg38/
- Breast Basal CEMT0035
- Breast Stromal CEMT0036
- Breast Luminal CEMT0037
- Breast Luminal Progenitor CEMT0038 #### MCF10A Fastq

```
ls ~/CourseData/EPI_data/module123/fastq
MCF10A.ATAC.chr19.R1.fastq.gz
MCF10A.ATAC.chr19.R2.fastq.gz
MCF10A.H3K27ac.chr19.R1.fastq.gz
```

```
MCF10A.H3K27ac.chr19.R2.fastq.gz
MCF10A.H3K27me3.chr19.R1.fastq.gz
MCF10A.H3K27me3.chr19.R2.fastq.gz
MCF10A.H3K4me3.chr19.R1.fastq.gz
MCF10A.H3K4me3.chr19.R2.fastq.gz
MCF10A.Input.chr19.R1.fastq.gz
MCF10A.Input.chr19.R2.fastq.gz
```

- MCF10A histone marks and input come courtesy of Dr.Hirst
- ATACseq data originates from GSM6431322 #### Triplicates

```
CourseData/EPI_data/module123/triplicates/triplicates.csv
```

```
CourseData/EPI_data/module123/triplicates/alignments:
MCF10A_H3K4me3_chr19.CondA.Rep1.bam      MCF10A_H3K4me3_chr19.CondB.Rep2.bam      MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep1.bam.bai  MCF10A_H3K4me3_chr19.CondB.Rep2.bam.bai  MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep2.bam      MCF10A_H3K4me3_chr19.CondB.Rep3.bam      MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep2.bam.bai  MCF10A_H3K4me3_chr19.CondB.Rep3.bam.bai  MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep3.bam      MCF10A_input_chr19.CondA.Rep1.bam        MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondA.Rep3.bam.bai  MCF10A_input_chr19.CondA.Rep1.bam.bai    MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondB.Rep1.bam      MCF10A_input_chr19.CondA.Rep2.bam        MCF10A_input_ch
MCF10A_H3K4me3_chr19.CondB.Rep1.bam.bai  MCF10A_input_chr19.CondA.Rep2.bam.bai    MCF10A_input_ch

CourseData/EPI_data/module123/triplicates/bigWig:
CondA.Rep1.bw  CondA.Rep2.bw  CondA.Rep3.bw  CondB.Rep1.bw  CondB.Rep2.bw  CondB.Rep3.bw

CourseData/EPI_data/module123/triplicates/peaks:
CondA.Rep1_peaks.narrowPeak  CondA.Rep3_peaks.narrowPeak  CondB.Rep2_peaks.narrowPeak
CondA.Rep2_peaks.narrowPeak  CondB.Rep1_peaks.narrowPeak  CondB.Rep3_peaks.narrowPeak
```

- triplicates were generated from MCF10A_H3K4me3 by choosing a list of exclusive peaks for condA and condB and subsampling replicates accordingly

Congratulations! You have completed Lab 3!

Chapter 7

Module 4: Whole-Genome Bisulfite Sequencing and Analysis

7.1 Lecture

7.2 Lab

7.2.1 1. Introduction

7.2.1.1 Description of the lab:

This module will cover the basics of Whole Genome Bisulfite-Sequencing (WGBS) data analysis including data visualization in IGV.

7.2.1.2 Objectives:

1) Learn how to align WGBS data using `Bismark`
2) Learn how to generate a methylation profile with `Bismark`
3) Learn how to open alignments and methylation profiles in the `IGV` genome browser
4) Learn how to perform a basic differential methylation analysis with `MethylKit`

7.2.1.3 Local software that we will use:

Before you begin, make sure you have the following programs ready in your local computer:

- A connection to the `EPI_2021` AWS instance
- An internet browser
- IGV
- R (or RStudio), *optional*

7.2.2 2. Mapping Tutorial

7.2.2.1 2.1 Getting Started

```
mkdir -p ~/workspace/module4
cd ~/workspace/module4
```

7.2.2.1.1 Prepare Directory for the Lab

```
GENOME=~/CourseData/EPI_data/module4/Homo_sapiens.GRCh38.chr19
export GENOME
```

7.2.2.1.2 Save Genome Location This will define a variable `$GENOME` that will simplify future commands.

7.2.2.2 Locate the Data for the Workshop

```
WGBS_DATA=~/CourseData/EPI_data/module4/data
export WGBS_DATA
```

This will define a variable `$WGBS_DATA` that will simplify future commands.

7.2.2.2.1 Check the Files Type the following command: `ls $WGBS_DATA`, what do you see?

You should see something similar to this

```
WGBS.A34002.137160.chr19.1.fastq.gz
WGBS.A34002.137160.chr19.2.fastq.gz
WGBS.A34002.137487.chr19.1.fastq.gz
WGBS.A34002.137487.chr19.2.fastq.gz
WGBS.A34002.137488.chr19.1.fastq.gz
WGBS.A34002.137488.chr19.2.fastq.gz
```

These are the files that will be used for the workshop. They contain a subset of WGBS reads from CEMT sample `CEMT0007`, which is a mammary gland epithelial cell line (more information here).

What do the ".1" and ".2" in the file names mean?

They represent the `read1` and `read2` of the paired end reads.

7.2.2.3 2.2 Map Data using Bismark

We will now process and map the reads using the Bismark WGBS aligner (more info here).

7.2.2.3.1 Map the first dataset using Bismark To simplify the work, we will process the datasets one at a time. To align the first dataset, do the following:

```
cd ~/workspace/module4
bismark --multicore 4 --bowtie2 $GENOME/genome/bismark_index \
    -1 $WGBS_DATA/WGBS.A34002.137160.chr19.1.fastq.gz -2 $WGBS_DATA/WGBS.A34002.137160.chr19.2.fa
```

What do all the options in the command mean? (Hint check the help by using `bismark --help`)

The `--multicore 4` option is to do multithreaded processing to improve speed.

The `--bowtie2` option is to use the mapping algorithm from bowtie2.

The `$GENOME/genome/bismark_index` specifies the location of the index for the reference genome to use. This uses the `$GENOME` variable we defined previously.

The `-1 $WGBS_DATA/WGBS.A34002.137160.chr19.1.fastq.gz` specifies the location of read 1. Idem for `-2` which specifies read 2. This uses the `$WGBS_DATA` variable we defined previously.

For more details, please refer to the Bismark user guide.

This step will take a few minutes to run for this reduced dataset. A dataset spanning a full genome will take *several hours*.

For your own datasets, make sure you have enough computing walltime to run the alignment.

While you wait for the results, ask any questions you have up to this point to the instructors.

7.2.2.3.2 Check files At the end of the alignment, you should have the following files saved into your workshop folder:

```
{:output} WGBS.A34002.137160.chr19.1_bismark_bt2_pe.bam  WGBS.A34002.137160.chr19.1_bis
```

Bismark offers a tool to generate an interactive HTML report for each sample, which we can now run to obtain the summary for our first sample.

```
bismark2report
```

This will produce an additional file called: `WGBS.A34002.137160.chr19.1_bismark_bt2_PE_report.htm`

Let's look at the report. We can read the text file directly on the command line:

```
less WGBS.A34002.137160.chr19.1_bismark_bt2_PE_report.txt
```

Or open the HTML report using your internet browser and the IP address of your AWS instance. Just click on the link to the HTML report and it should open.

What was the mapping efficiency? What percent of C's were methylated in CpG context?

According to the report:

```
...
Mapping efficiency:     92.4%
...
C methylated in CpG context:    57.4%
C methylated in CHG context:    0.6%
C methylated in CHH context:    0.5%
C methylated in unknown context (CN or CHN):    3.5%
...
```

You can also look at plot the `Cytosine Methylation` section in the interactive HTML report.

Close the report by pressing `q`.

7.2.2.3.3 Prepare files for loading in IGV We need to sort the `bam` file and prepare an index so we will be able to load it in IGV. We will use the program `samtools` for this.

```
samtools sort WGBS.A34002.137160.chr19.1_bismark_bt2_pe.bam -o WGBS.A34002.137160.chr19.1_bismark
samtools index WGBS.A34002.137160.chr19.1_bismark_bt2_pe_sorted.bam
```

7.2.2.3.4 Check Files At the end, you should have the following files:

```
WGBS.A34002.137160.chr19.1_bismark_bt2_pe_sorted.bam
WGBS.A34002.137160.chr19.1_bismark_bt2_pe.bam
WGBS.A34002.137160.chr19.1_bismark_bt2_pe_sorted.bam.bai
WGBS.A34002.137160.chr19.1_bismark_bt2_PE_report.txt
```

7.2.2.4 2.3 Repeat Alignment for All Datasets

How would you repeat the alignment with the other datasets?

This is the command to run bismark on the two other samples:

```
cd ~/workspace/module4

bismark --multicore 4 --bowtie2 $GENOME/genome/bismark_index \
    -1 $WGBS_DATA/WGBS.A34002.137487.chr19.1.fastq.gz -2 $WGBS_DATA/WGBS.A34002.137487.chr19.2.fa

bismark --multicore 4 --bowtie2 $GENOME/genome/bismark_index \
    -1 $WGBS_DATA/WGBS.A34002.137488.chr19.1.fastq.gz -2 $WGBS_DATA/WGBS.A34002.137488.chr19.2.fa
```

Remember, for the command to work, both `$GENOME` and `$WGBS_DATA` need to be defined.

Also, if you want to generate the HTML reports, you can run `bismark2report`. Bismark also has a "summary" that produces a report for all samples (`bismark2summary`), we can run both by doing the following:

```
bismark2report ; bismark2summary
```

After checking the reports, we can run the commands to prepare the samples for IGV (sort and index):

```
samtools sort  WGBS.A34002.137487.chr19.1_bismark_bt2_pe.bam -o WGBS.A34002.137487.chr
samtools index WGBS.A34002.137487.chr19.1_bismark_bt2_pe_sorted.bam

samtools sort  WGBS.A34002.137488.chr19.1_bismark_bt2_pe.bam -o WGBS.A34002.137488.chr
samtools index WGBS.A34002.137488.chr19.1_bismark_bt2_pe_sorted.bam
```

7.2.2.5 2.4 Load Data and Explore using IGV

While you wait for the previous steps to finish executing, it is a good idea to begin exploring the alignments.

7.2.2.5.1 Copy Files to Your Local Computer to View in IGV Retrieve the files called `WGBS.A34002.137160.chr19.1_bismark_bt2_pe_sorted.bam` and `WGBS.A34002.137160.chr19.1_bismark_bt2_pe_sorted.bam.bai` from the server using your internet browser and the IP address of your AWS instance.

Optionally, if you are not using Putty (i.e. if you are using a Linux or Mac computer) you can also retrieve the files directly using the terminal and `scp` using the commands below.

Optional download option (click here)

```
scp -i CBW.pem ubuntu@00.00.00.0:~/workspace/module4/WGBS.A34002.137160.chr19.1_bismarl
scp -i CBW.pem ubuntu@00.00.00.0:~/workspace/module4/WGBS.A34002.137160.chr19.1_bismarl
```

Remember that for the commands above to work, you need to be in the directory with the `CBW.pem` (or `CBW.cer`) file you downloaded from AWS when creating your instance.

7.2.2.5.2 Launch IGV on your computer If you haven't installed it yet, please get it here IGV download.

Make sure that the human genome is selected in the top left corner. It should read: **Human (hg38)**.

Load your **sorted** `bam` file in IGV using `File -> Load from file`. *For this to work, you need to have the index file (`.bai`) in the same location as the `bam` file.*

Now go to the following location:

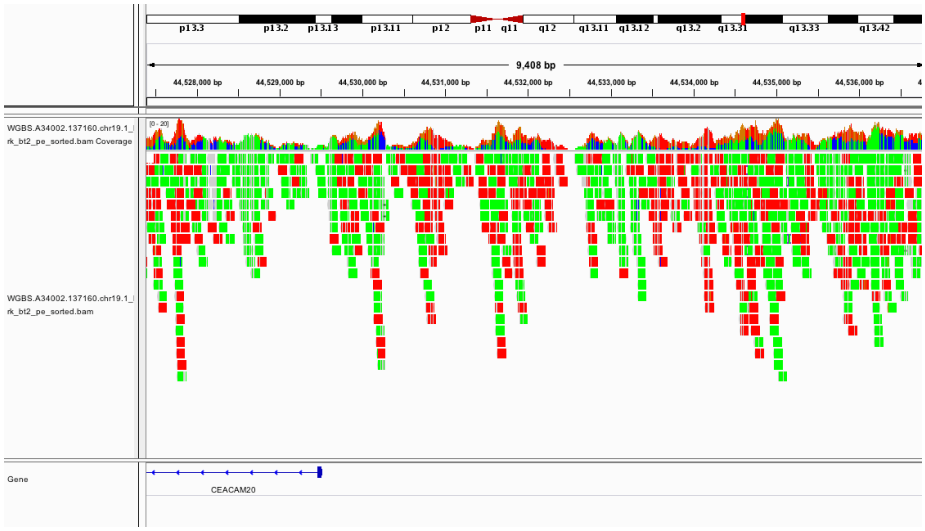
```
chr19:43,375,889-45,912,052
```

And zoom in until you see something.

For instance, try the following window:

```
chr19:44,527,387-44,536,873
```

You should see something like this:



If it looks different, can you change the way the colors are displayed?

Which section of which chromosome is covered by this dataset?

Can you see any interesting patterns in the coverage?

7.2.2.6 2.5 Generate Methylation Profiles

So far we have only mapped the reads using Bismark. We can generate methylation profiles using the following command:

```
cd ~/workspace/module4
```

```
bismark_methylation_extractor --cytosine_report WGBS.A34002.137160.chr19.1_bismark_bt2_pe.bam --g
```

The `--cytosine_report` option creates a `CpG_report` table summarizing key data for each `CG` position in the genome, which will be useful later (see section 4.2).

How would you do the same for the other replicates?

These are the commands that you should use:

```
cd ~/workspace/module4

bismark_methylation_extractor --cytosine_report WGBS.A34002.137487.chr19.1_bismark_bt2_
bismark_methylation_extractor --cytosine_report WGBS.A34002.137488.chr19.1_bismark_bt2_
```

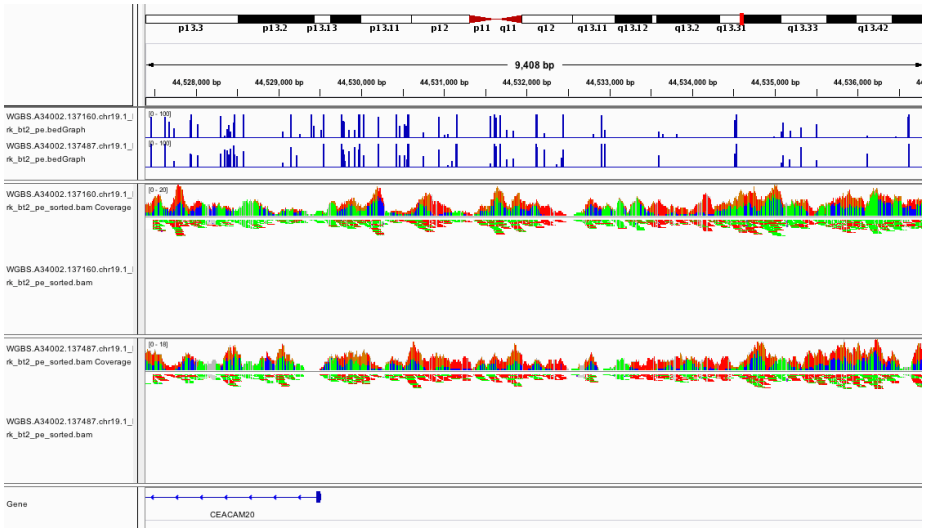
Download *all* the files produced so far to your local computer using your internet browser.

While you wait for all the steps and downloads to finish, you can ask the instructors any questions you might have up until this point.

Load all the downloaded files in IGV using `File -> Load from file`.

Please be aware that if you just try to open all your files on IGV you might get a warning/error message mentioning that you are trying to open an unsorted BAM. If that is the case, ignore the message, but make sure that you are opening the `sorted.bam` so you can visualize your results.

At this point, if you load the region `chr19:44,527,387-44,536,873` you should see something like



This promoter looks to be hypomethylated.

Can you find a promoter that is hypermethylated?

How about `chr19:45,637,715-45,657,380`?

How would you look for a CpG island using this view of the data?

7.2.3 3. Differential Methylation Analysis in MethylKit

The following section will use the Bioconductor package `methylKit` to do a differential methylation analysis. You can do it in your own computer (if you have installed `R` and `methylKit`) or in the AWS instance.

To install `methylKit` locally on your computer, make sure you have a recent version of `R` and follow the instructions in this page.

7.2.3.1 3.1 Load R and MethylKit

If you are working in AWS, you will need to load `R`. The image we provide already has the libraries we need.

To launch `R` simply type the following to your terminal:

```
cd ~/workspace/module4
R
```

If you did this properly, the following message will be displayed and your prompt will change from **ubuntu@ip-00-00-00-0:~/workspace/module4$** to **>**:

"'{:output} R version 4.2.3 (2023-03-15) – "Shortstop Beagle" Copyright (C) 2023 The R Foundation for Statistical Computing Platform: x86_64-pc-linux-gnu (64-bit) …

```
Once you have successfully launched `R`, you can load `methylKit` with the following co

```{}
library("methylKit")
```

7.2.3.2 3.2 Import the Alignment Data into methylKit

7.2.3.2.1 Process Bismark Alignments To read the alignment data into `methylKit`, run the following command:

```
methRaw.160 = processBismarkAln( location = "WGBS.A34002.137160.chr19.1_bismark_bt2_pe_
                          sample.id="A34002.137160", assembly="hg38",
                          read.context="CpG", save.folder="methylkit")
```

This command will import the data into a format that is readable by `methylKit`. At the same time, it will save two files under the `methylkit` directory with the information so that it is easy to load again at any time:

```
methylkit/A34002.137160_CpG_conversionStats.txt
methylkit/A34002.137160_CpG.txt
```

If everything goes well and you see the files, do the same for the other two samples:

```
methRaw.487 = processBismarkAln( location = "WGBS.A34002.137487.chr19.1_bismark_bt2_pe_
                          sample.id="A34002.137487", assembly="hg38",
                          read.context="CpG", save.folder="methylkit")

methRaw.488 = processBismarkAln( location = "WGBS.A34002.137488.chr19.1_bismark_bt2_pe_
                          sample.id="A34002.137488", assembly="hg38",
                          read.context="CpG", save.folder="methylkit")
```

```

7.2.3.2.2 Create a MethylKit Object Now that all the samples have been read with `methylKit`, you can create a file list to make it easier to load the full dataset as a `methylkit` object. For the purposes of this tutorial, we will consider that samples belong to two experimental groups: `A34002.137160` as the control group (treatment = 0) and `A34002.137487` & `A34002.137488` as the treatment group (treatment = 1). We use the `methRead()` function to create our object, as shown below:

```
file.list = list( file.path("methylkit", "A34002.137160_CpG.txt"),
                  file.path("methylkit", "A34002.137487_CpG.txt"),
                  file.path("methylkit", "A34002.137488_CpG.txt") )

myobj = methRead(file.list,
           sample.id=list("A34002.137160","A34002.137487","A34002.137488"),
           assembly="hg38",
           treatment=c(0,1,1),
           context="CpG",
           mincov = 10
           )
```

What do all the options in the `methRead()` command mean?

- `file.list` object points to the location of the input data in a MethylKit format.
- `sample.id` points to a list with the appropriate sample name for each file.
- `assembly` specifies which build of the human reference genome is used.
- `treatment` specifies which sample belongs to each experimental group.
- `context` specifies the methylation context.
- `mincov` specifies the minimum coverage required to be included in the object.

For more details, please refer to the MethylKit user guide .

If the files were loaded properly, you can check the object you just created by running the following command:

```
myobj
```

Which should output the following message followed by previews of the contents of the object:

```
methylRawList object with 3 methylRaw objects
...
```

You can also get basic statistics on your object by using the following command:

```
getMethylationStats(myobj[[2]],plot=FALSE,both.strands=FALSE)
```

7.2.3.3 3.3 Find Differentially Methylated Regions with methylKit

7.2.3.3.1 Merge Samples Before doing any additional analysis, `methylKit` needs to determine which methylated bases have sufficient coverage in all samples so they can be compared. To do that, the samples should be merged with the `unite()` function. This function has a parameter `destrand=` that is turned off by default. We will set the `destrand` option to `TRUE` which will merge the coverage of both strands. When doing your own analyses, be aware that for some kinds of methylation analyses (such as CpH methylation) results are strand-specific, so this option should be used carefully.

```
meth = unite(myobj, destrand=TRUE)
```

7.2.3.3.2 Perform Differential Methylation Analysis The standard function for Differential Methylation Analysis on `methylKit` is `calculateDiffMeth()`. It takes any **merged** methylkit object as input. Depending on the number of replicates, it uses either Fisher's exact or logistic regression to calculate P-values. It also, automatically produces Q-values, which are a kind of adjusted P-value. To use it with the results we obtained before, run the following command:

```
myDiff = calculateDiffMeth(meth)
```

To check the output, just type `myDiff` and read the summary. If you want an example of the output, check the solution below.

This is what the output looks like:

```
{:output} methylDiff object with 2941 rows --------------     chr
start      end strand       pvalue     qvalue  meth.diff 1 chr19
42002896 42002896      + 3.271268e-01 0.69569299   8.951407 2
```

```
chr19 42002978 42002978      + 1.912989e-01 0.60732656 -21.666667
3 chr19 42007251 42007251      + 6.999764e-05 0.03228847 -55.681818
4 chr19 42007255 42007255      + 3.958578e-01 0.75196047 -11.835106
5 chr19 42007283 42007283      + 8.451850e-01 0.91347038  -2.457757
6 chr19 42007314 42007314      + 9.102723e-01 0.92865750  -1.604278
-------------- sample.ids: A34002.137160 A34002.137487 A34002.137488
destranded TRUE  assembly: hg38  context: CpG  treament: 0 1 1
resolution: base
```

To filter results by their statistical significance, `methylKit` provides the `getMethylDiff()` function which allows you to extract only the deferentially methylated CpG's that meet a specific Q-value threshold. Additionally, it is also possible to specify whether to keep `hypo` or `hyper` methylated CpG's only. Finally, the `bedgraph()` function allows you to save the the `methylDiff` object into a BedGraph file so you can open it with your genome browser of choice. Let's create two BedGraph files with hypo and hyper methylated CpG's with a Q-value below 0.05 based on the data above:

```
myDiff.hyper = getMethylDiff(myDiff,qvalue=0.05,difference=10,type="hyper")
bedgraph(myDiff.hyper, file.name = "hyper.CpG.bedGraph", col.name = "qvalue")

myDiff.hypo = getMethylDiff(myDiff,qvalue=0.05,difference=10,type="hypo")
bedgraph(myDiff.hypo, file.name = "hypo.CpG.bedGraph", col.name = "qvalue")
```

Two new files should appear now in your workshop folder:

```
{:output} ~/workspace/module4/hyper.CpG.bedGraph ~/workspace/module4/hypo.CpG.bedGraph
```

7.2.3.3.3 Bin Results to Obtain Differentially Methylated Regions
By default, `methylKit` will compute results with an individual CpG resolution. To get Differentially Methylated Regions (**DMR**), you have to bin your results first, using a window size of your choice. The function to do this is `tileMethylCounts()`, which takes a regular methylkit object as input. In this case, we will create 1000bp bins using the following command:

```
tiles = tileMethylCounts(myobj,win.size=1000,step.size=1000,cov.bases = 10)
```

As with CpG level results, samples need to be merged before the analysis can continue:

```
meth.tiles = unite(tiles, destrand=TRUE)
```

Now, we will use the `calculateDiffMeth()` and `getMethylDiff()` functions to get the DMRs.

Do you know how to do it, based on the information above?

Based on the number of differentially methylated CpGs you found above, do you anticipate many statistically significant DMRs in your analysis?

Use the following commands to perform a `DMR` analysis:

```
myDiff.tiles = calculateDiffMeth(meth.tiles)

myDiff.tiles.hyper = getMethylDiff(myDiff.tiles,qvalue=0.1,difference=10,type="hyper")
bedgraph(myDiff.tiles.hyper, file.name = "hyper.DMR.bedGraph", col.name = "qvalue")

myDiff.tiles.hypo = getMethylDiff(myDiff.tiles,qvalue=0.1,difference=10,type="hypo")
bedgraph(myDiff.tiles.hypo, file.name = "hypo.DMR.bedGraph", col.name = "qvalue")
```

Using the navigation pane, download the bedGraph files you just produced and try to open them with IGV.

Do the statistical results match what you had seen before when exploring the data?

What interesting genomic features are found close to the DMRs? What could this mean?

7.2.4 4. Differential Methylation Analysis in DSS

The following section will use the Bioconductor package `DSS` to do a differential methylation analysis. You can do it in your own computer (if you have installed `R` and `DSS`) or in the AWS instance.

To install `DSS` locally on your computer, make sure you have a recent version of `R` and follow the instructions in this page.

7.2.4.1 4.1 Load R and DSS

If you just did the previous section, you might not need to load `R` again. Otherwise, please launch `R` using the instructions in section 3.1.

Once you have successfully launched `R`, you can load `DSS` with the following command:

```
library("DSS")
```

7.2.4.2 4.2 Import the Methylation Data into DSS

7.2.4.2.1 Process Bismark CpG Reports The `DSS` library expects input data to be already summarized into the following columns for each CG position in the genome:

- chromosome number (`chr`),
- genomic coordinate (`pos`),
- total number of reads (`N`),
- and, number of reads showing methylation (`X`).

Fortunately, Bismark already produced a table (`CpG_report.txt`) with most of that information when we ran the `bismark_methylation_extractor` command (see section 2.5 to review this step). Therefore, we can import the `CpG_report` table into R and reshape it into the proper input for `DSS`.

First, we will load the `CpG_report.txt` table for sample `137160` to our R environment and save it as a variable called `CpG.report.160`. To do this, we will use the base `R` function `read.table` and name the columns as follows:

- `chr`,
- `pos`,
- `strand`,
- `X` (num. of reads showing methylation at this position),
- `C` (num. of reads *without* methylation in this position),
- `C_context` (2-base context at this position),
- `tri_context` (3-base context at this position)

The full command is as follows:

```
CpG.report.160 <- read.table("WGBS.A34002.137160.chr19.1_bismark_bt2_pe.CpG_report.txt", header =
```

Next, we need to calculate the total number of reads for each position by adding up the ones with and without methylation (columns `X` and `C` in the table we just imported). We will name this new column `N` to follow the `DSS` convention. Finally, we will save the input table for DSS as `CpG.DSS.table.160` and make sure it is in ascending order by position with the `setorder` command. You can find the full commands below:

```
CpG.report.160["N"] <- CpG.report.160["C"] + CpG.report.160["X"]
CpG.DSS.table.160 <- CpG.report.160[c("chr", "pos", "N", "X")]
```

Can you repeat the above process for the other two samples?

Use the following commands to import the remaining `CpG_report` tables, as well as reformatting them into appropriate `DSS` input.

```
CpG.report.487 <- read.table("WGBS.A34002.137487.chr19.1_bismark_bt2_pe.CpG_report.txt
CpG.report.487["N"] <- CpG.report.487["C"] + CpG.report.487["X"]
CpG.DSS.table.487 <- CpG.report.487[c("chr", "pos", "N", "X")]

CpG.report.488 <- read.table("WGBS.A34002.137488.chr19.1_bismark_bt2_pe.CpG_report.txt
CpG.report.488["N"] <- CpG.report.488["C"] + CpG.report.488["X"]
CpG.DSS.table.488 <- CpG.report.488[c("chr", "pos", "N", "X")]
```

Hint: If you want to check all the `DSS` input tables are constructed properly, you can take a peek at the "top" of each table with the `head()` command in `R`. For example, you should see the following if you did all the previous steps correctly:

```
head(CpG.DSS.table.160)
```

```
{:output} chr   pos N X 1 chr19 60119 0 0 2 chr19 60120 0 0 3
chr19 60172 0 0 4 chr19 60173 0 0 5 chr19 60183 0 0 6 chr19 60184
0 0
```

The same command for the other two samples will give the same result, because there are no reads aligning to the beginning of chromosome 19 in our dataset.

Next, we will create the `BS` object that `DSS` will use by using the `makeBSseqData` command and the tables we just created:

```
BSobj <- makeBSseqData( list(CpG.DSS.table.160, CpG.DSS.table.487, CpG.DSS.table.488),
     c("sample.160","sample.487", "sample.488") )
```

Notice how we are labeling the 3 samples with the names `c("sample.160","sample.487", "sample.488")` as part of this command. It is likely that you will receive a warning message when you run the command above indicating that the CG sites are not ordered. We will ignore it for now, in this case it should not impact the analysis.

If the object was constructed properly, we can inspect it's properties by calling the variable name directly `BSobj`, which will output.

```
{:output} An object of type 'BSseq' with   2113330 methylation
loci   3 samples has not been smoothed All assays are in-memory
```

7.2.4.3 4.3 Find Differentially Methylated Regions with DSS

The `DSS` library operates under different statistical assumptions than `methylKit`, which include a smoothing of methylation data to improve methylation analysis. By smoothing the data, `DSS` will attempt to correct the methylation percentage of a CpG site using the context of nearby sites. Additionally, it will then estimate the dispersion of the sites and perform a Wald statistical test to allow for comparsion across samples. These three steps are done in one single command called `DMLtest` which will take our `BSobj` as input.

We will run command is done as follows and save the results in a variable called `dmlTest`:

```
dmlTest <- DMLtest(BSobj,
  group1=c("sample.160"),
  group2=c("sample.487", "sample.488"),
  smoothing=TRUE)
```

Notice how we defined the two experimental groups in the `DMLtest` command, using the we defined when creating `BSobj`. When the test finishes running, we can extract the significant Differentially Methylated Loci (DML) using the `callDML` command and our defined p-value threshold:

```
dmls = callDML(dmlTest, p.threshold=0.05)
```

Extracting Differentially Methylated Regions (DMR) works similarly, with the `callDMR` command. One important difference between `methylKit` and `DSS` is that the latter does not work using bins. Instead, `DSS` will estimate the length of a DMR based on the dispersion of individual CpGs. Therefore, contrary to DMRs calculated with methylKit, these regions will all have different lengths.

To ensure that DMRs we call in this workshop are comparable to what we calculated with methylKit, we will set the minimum length to 500bp. The `callDMR` command will look like this:

```
dmrs = callDMR(dmlTest, minlen = 500, p.threshold=0.05)
```

The `DSS` library will group all DMR results into a single table, making no distinction between hypo and hyper methylated regions. You can view the results by running calling the `dmrs` variable.

Look at the `DSS` differential methylation results and compare them to what was obtained by `methylKit`. Do you notice any important differences? Are there any overlaps?

To save the results of your `DSS` analysis as a `CSV` file, use the base R command `write.csv`. Unfortunately, there is no native `DSS` command to export the results into a BedGraph file, and converting the table outptut to this format is outside the scope of this workshop. We encourage you to find tools that do this after the workshop if you want to do a deeper comparsion between the `DSS` and `methylKit` results in your own datasets.

```
write.csv(dmls, file="DSS.DML.table.csv")
write.csv(dmrs, file="DSS.DMR.table.csv")
```

After saving the `DSS` results as CSV, remember to download them to your computer, where you can open them in the file explorer, or even a spreadsheet to improve visualization and filtering.

7.2.4.4 Congrats, you're done!

You can quit R using the `quit()` or `q()` command. Remember to stop your AWS instance after this lab to avoid unnecessary costs.

Once you are finished make sure you download all the files you need and continue exploring on IGV.

Congratulations! You have completed Lab 4!

Chapter 8

Module 5: Downstream Analysis and Online Tools

8.1 Lecture

8.2 Lab

8.2.1 Introduction

8.2.1.1 Description of the Lab

We will now explore some of the tools that were covered in the lecture for module 5.

In this lab, we will:

- Learn how to use an online resource, the IHEC Data Portal, to fetch feature tracks of interest.
- Explore ChIP-Seq peak prediction files (in bed format) to discover motifs using HOMER.
- Use an IHEC dataset with the GREAT GO enrichment tool to perform functions prediction.

8.2.1.2 Local Software That will be Needed

- Tool to connect to your remote terminal session (e.g. Putty in Windows)
- A web browser
- The IGV genome browser

8.2.1.3 Prepare Directory for Module 5

From your command line terminal, go to your workspace folder.

```
cd ~/workspace
```

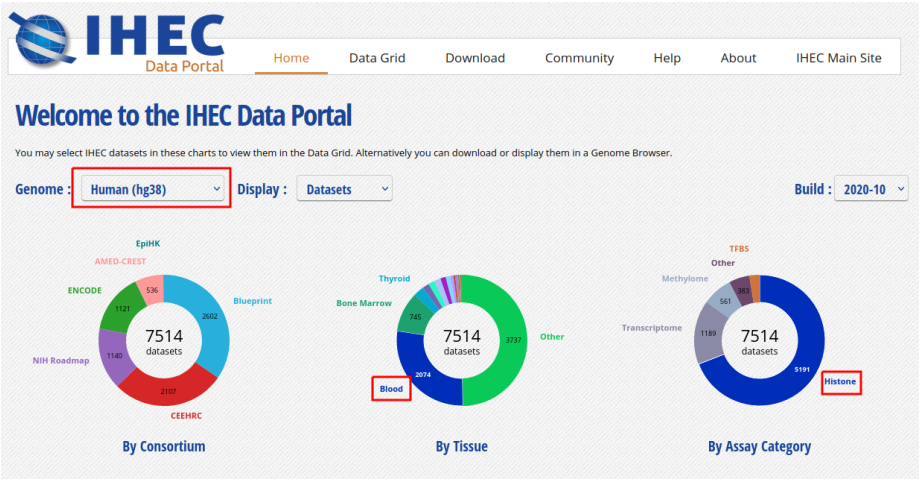
- If it exists, remove any `module5` directory in your home directory with the "rm" command
- Create a new `module5` directory
- Go to that directory

```
rm -rf module5
mkdir module5
cd module5
```

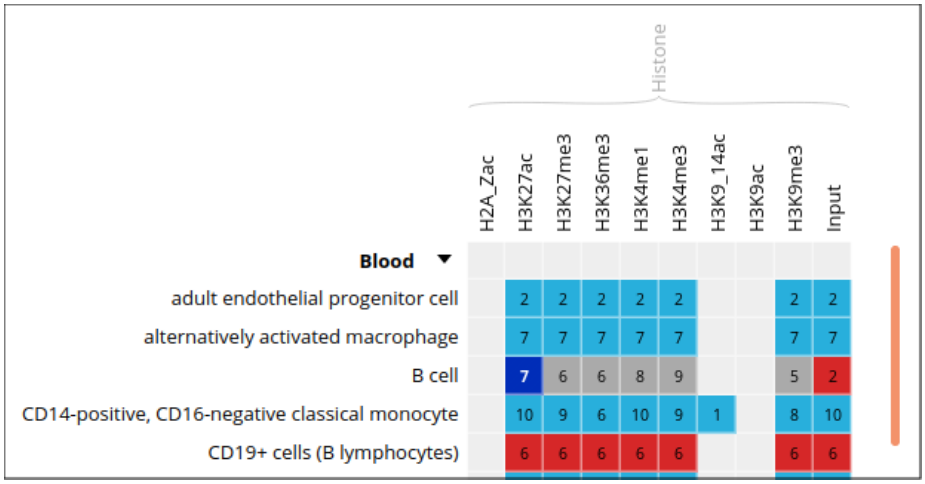
8.2.2 IHEC Data Portal

8.2.2.1 Exploring Available Datasets

- Open a web browser on your computer, and load the URL http://epigenomesportal.ca/ihec
- In the Overview page, click on the "View all" button, below the pie charts
- You will get a grid with all available datasets for IHEC Core Assays, on the hg38 assembly
  - You can filter out visible datasets in the grid using the filtering options at the right of the grid
- Go back to the Overview page (`Home` on the top menu), and select the following categories of datasets: On the "hg38" reference genome, "Histone" experiments for the "Blood" cell type. Click on `View selected`

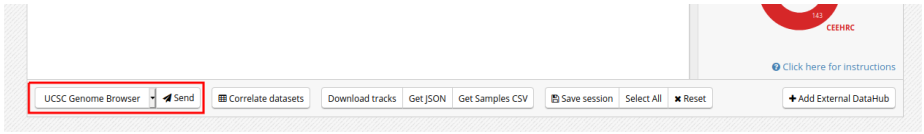


- Only these categories will now get displayed in the grid. Expand the "Blood" category by clicking on the black triangle, and select the following grid cell:



8.2.2.2 Visualizing the Tracks

- Click on the "Send" button for the UCSC Genome Browser, at the bottom of the grid



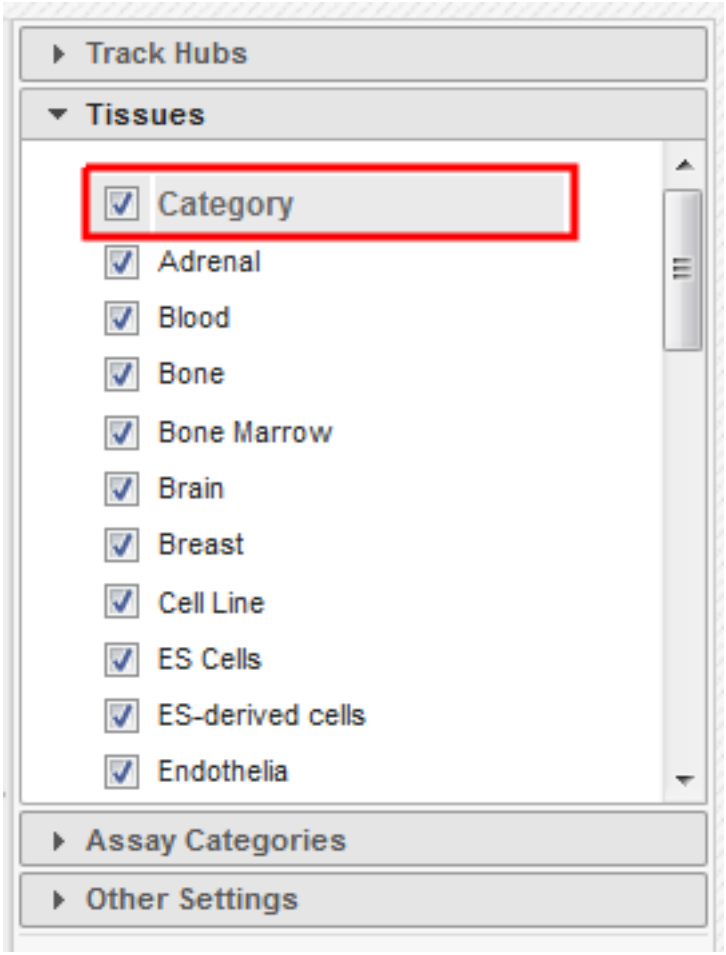
- You can see that the datasets are being displayed on the UCSC Genome Browser. These are all peaks and signal for the chosen blood H3K27ac ChIP-Seq datasets. In the Genome Browser, you can expand the tracks by changing visibility from "pack" to "full" and clicking the "Refresh" button

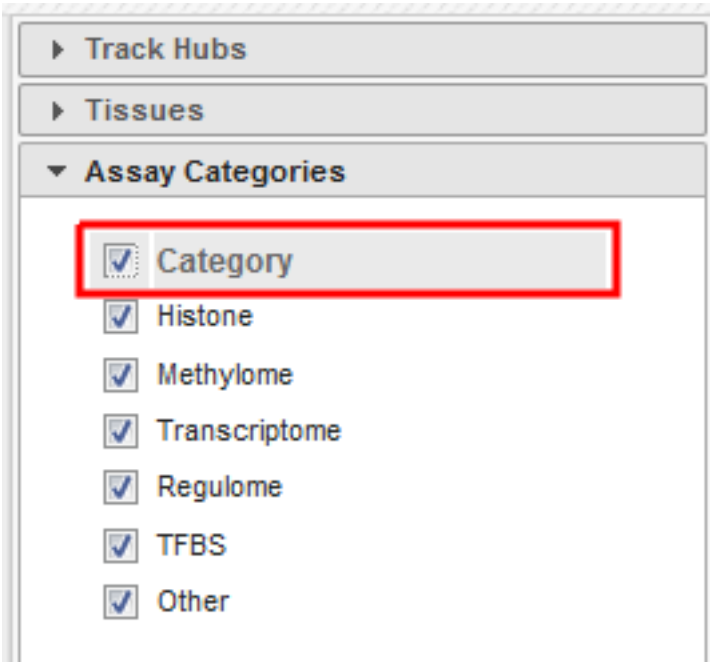
- You can also download these tracks locally for visualization in IGV
  - Go back to the IHEC Data Portal tab
  - Click on the "Download tracks" button at the bottom of the grid
  - Use the download links to download a few of the available tracks
  - Open them in IGV

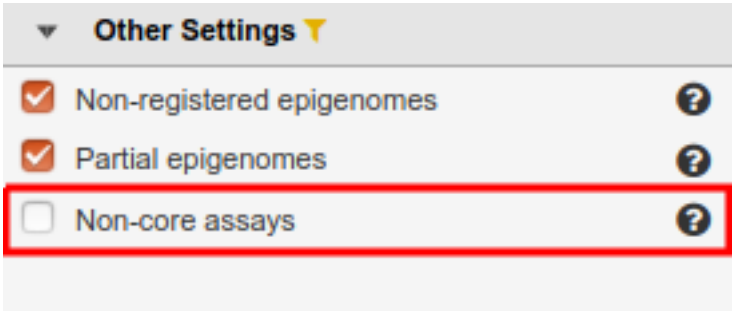
8.2.2.3 Tracks Correlation

You can get a whole genome overview of the similarity of a group of tracks by using the Portal's correlation tool.

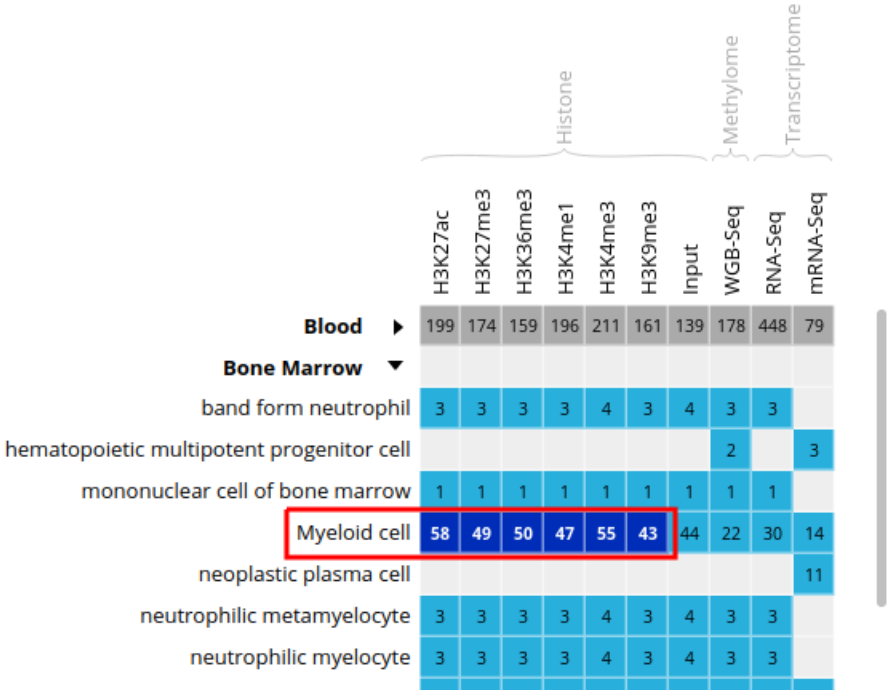
- Back on the Data Grid tab of your browser, from the filters at the right of the grid, add back datasets for all tissues and all assay types. You can select all checkboxes at once by click on the top checkbox, next to "Category". Also remove non-core assays if it is selected







- Select all ChIP-Seq marks for the cell type “Myeloid cell”, under the “Bone Marrow” category. The first 6 columns should be selected

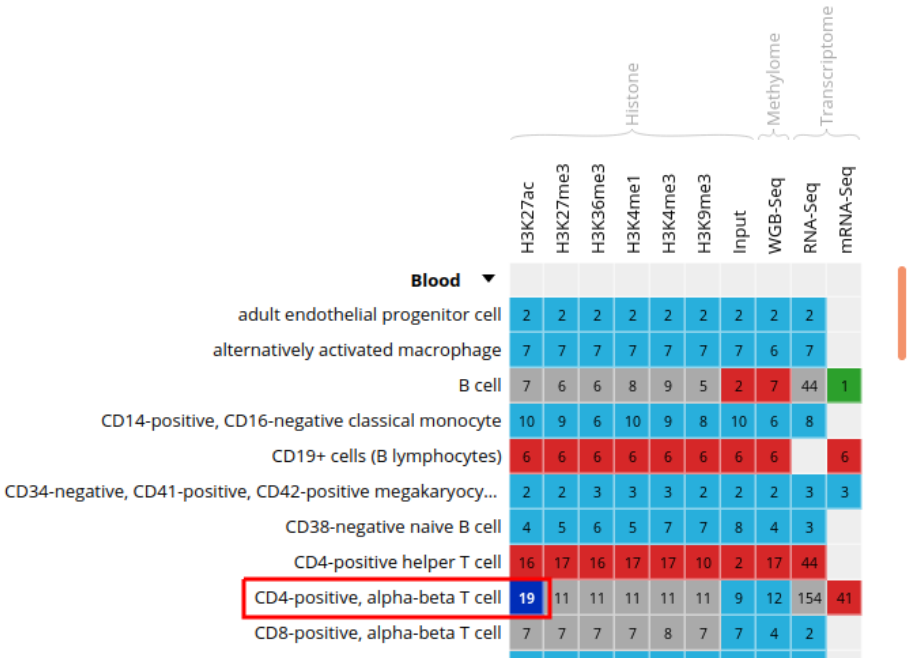


- At the bottom of the grid, click on the button “Correlate datasets”

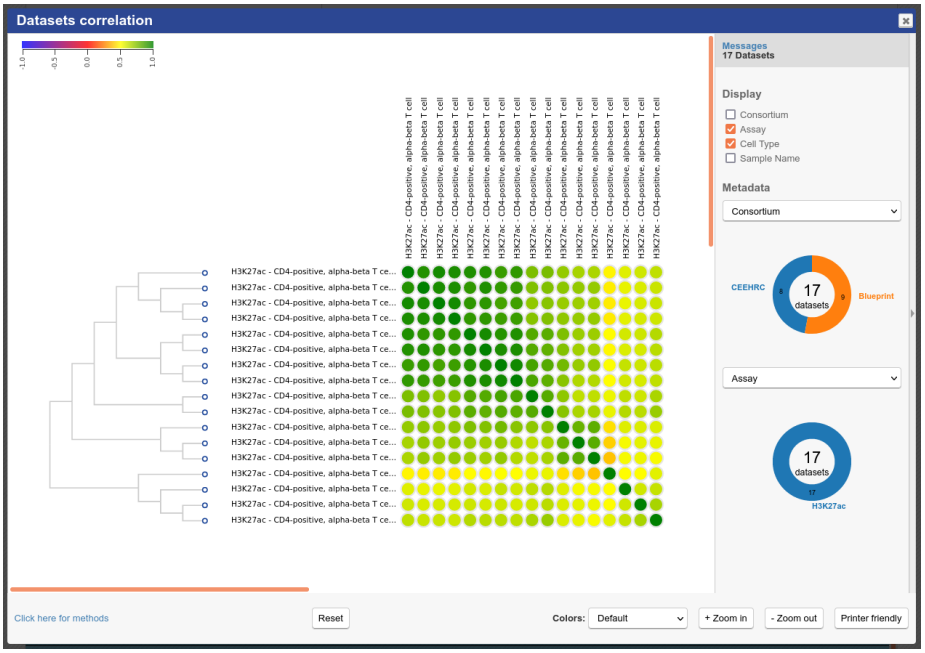
- You will see that similar tracks (e.g. those of the same assay, seem to correlate nicely. You can zoom in the view with the mouse scrolling wheel, or with the buttons at the lower right corner of the popup

- You can also use the correlation tool to assess whether datasets that are supposed to be similar actually are

  - Close the correlation popup window with the top right "X" button
  - Reset grid selection with the "Reset" button at the bottom of the grid
  - Click on the grid cell for cell type "CD4-positive, alpha-beta T cell", under the "Blood" category, and assay "H3K27ac"
  - Click on "Correlate datasets"
  - One dataset seems to be an outlier. This could be, for instance, a problem with the quality of the dataset, or the underlying metadata can indicate that something is different (disease status or some other key element)



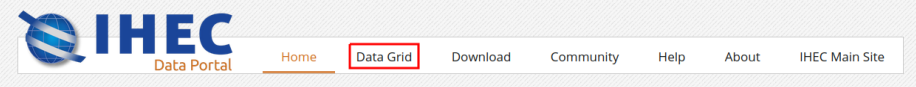
You should get something like this:



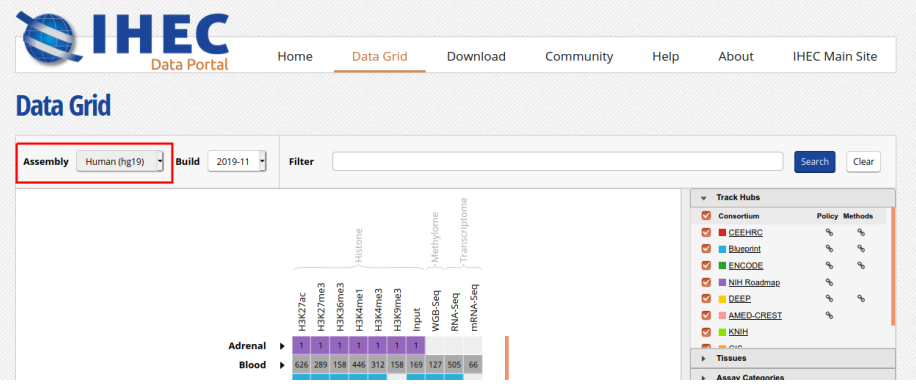
8.2.3 Predicting Motifs with HOMER

We will now attempt to detect motifs in peak regions for transcription factor binding sites using HOMER

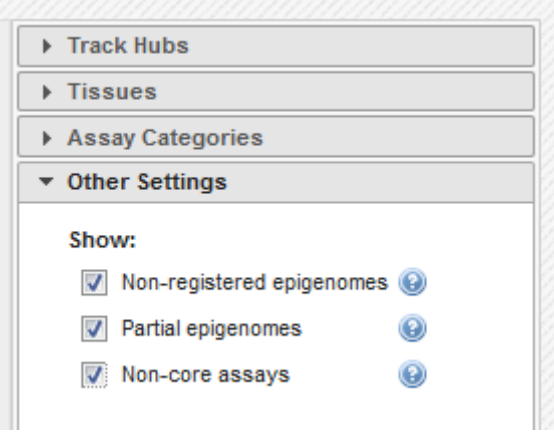
- Reset to the default IHEC Data Portal view by clicking "Data Grid" in the top bar

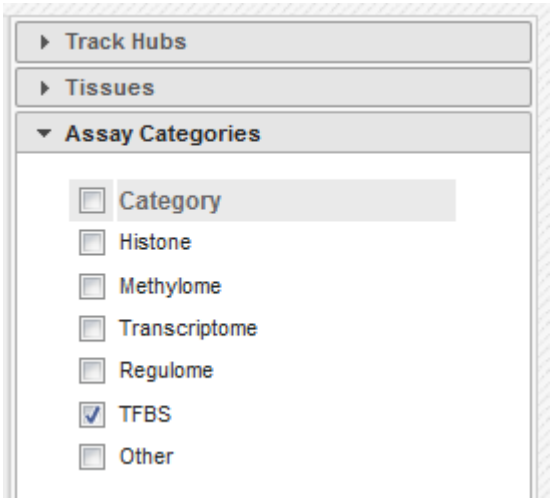


Choose Assembly `hg19`.

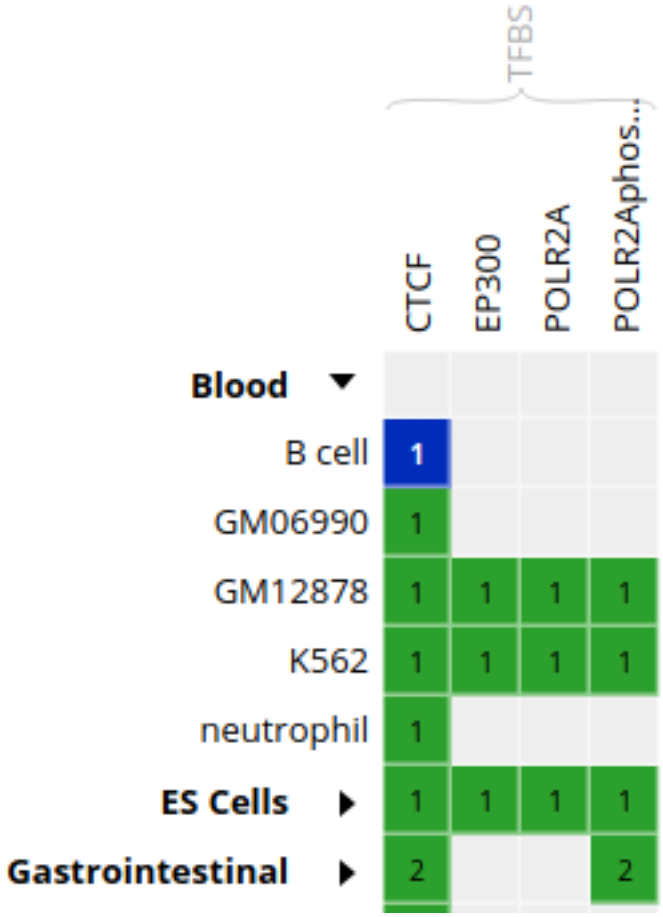


- In the filters to the right of the grid, activate non-core IHEC assays, and display only Transcription Factor Binding Sites (`TFBS`) assays

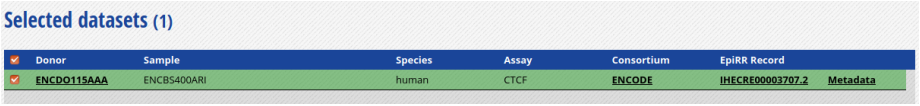




- In the grid, select ENCODE datasets for the `CTCF` assay and the `B cell` cell type



- Go to the track list at the bottom of the grid and select only the dataset for sample "ENCBS400ARI"



- You can get the URL to the track you want by clicking on the "Download tracks" button at the bottom of the grid Here, we're interested in `https://epigenomesportal.ca/tracks/ENCODE/hg19/84840.ENCODE.ENCBS400ARI.CTCF.peak_calls.bigB` This file contains peaks that were called out of the TFBS ChIP-Seq experiment
- Useful tip: You can get the full set of metadata about ENCODE experiments and samples by consulting the ENCODE

portal, and searching for a given sample name. In this case: `https://www.encodeproject.org/biosamples/ENCBS400ARI/`
- Open your AWS terminal session, create a directory for our HOMER-related files, and go into it. Then, download the BigBed file

```
mkdir homer
cd homer
wget https://epigenomesportal.ca/tracks/ENCODE/hg19/84840.ENCODE.ENCBS400ARI.CTCF.peak
```

- UCSC provides a set of file format conversion tools, such as `bigBedToBed`, which converts a binary bigbed file to its plain text file equivalent. Some of these tools have been pre-installed on your AWS image
- Convert the bigBed file into a bed file using the UCSC set of tools

```
bigBedToBed 84840.ENCODE.ENCBS400ARI.CTCF.peak_calls.bigBed 84840.ENCODE.ENCBS400ARI.C
```

- Prepare an output directory for HOMER, and a genome preparsed motifs directory

```
mkdir output
mkdir preparsed
```

- Run the HOMER software to identify motifs in the peak regions
  - -p 4 indicates to use 4 cores.
  - -S 15 tells Homer to find 15 motifs, instead of the default 25, for execution speed purposes.
  - You can get the full list of parameters here: http://homer.ucsd.edu/homer/ngs/peakMotifs.html

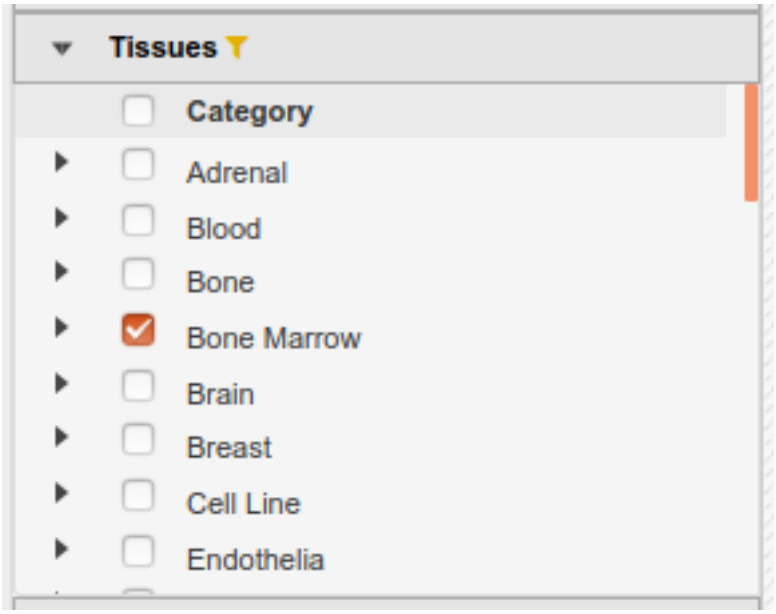
```
findMotifsGenome.pl 84840.ENCODE.ENCBS400ARI.CTCF.peak_calls.bed hg19 output -preparse
```

- HOMER takes a while to execute for a whole genome track like this. Expect this job to take about 30 minutes of runtime, with your current setup. In the meantime, we will explore the GO terms enrichment tool GREAT

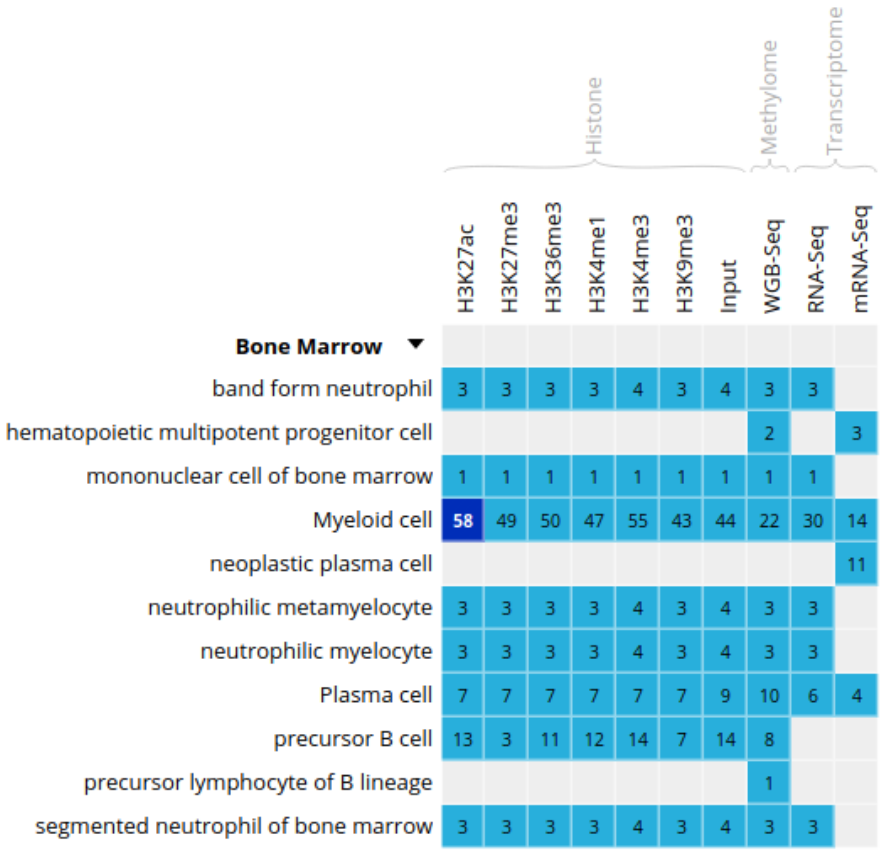
8.2.4 Looking for GO Terms Enrichment with GREAT

Next, we will try to identify GO terms connected to ChIP-Seq peaks calls using GREAT. We need `bed` files to use the GREAT portal. We will do the conversion from a `bigBed` file to a `bed` file on our AWS session

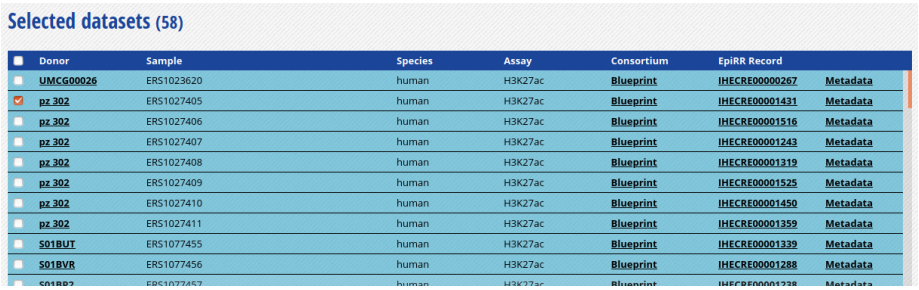
- In the IHEC Data Portal, go back to the default grid page (by clicking on Data Grid in the top bar). For assembly `Human (hg38)`, filter the tissues list to keep only "Bone Marrow" tissues



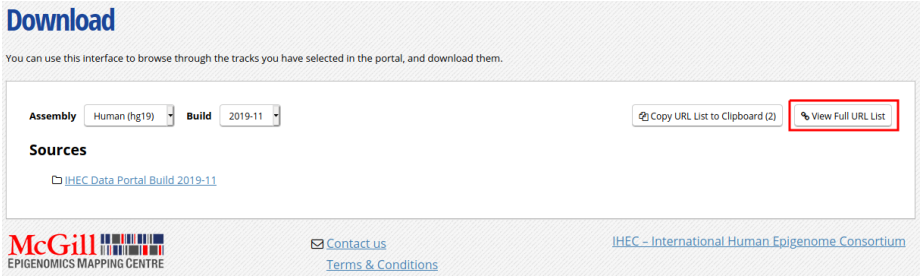
- Select the datasets for cell type "Myeloid cell" and assay H3K27ac



- For this exercise, we will download only one of the bigbeds for available datasets. Pick up the dataset below, for sample `ERS1027405`:



- Click "Download tracks" at the bottom of the grid
- On the download page, click on `View Full URL List`. This will give you a text list with all tracks of interest Copy the link to this page in your clipboard, using the address provided in your browser's URL bar



- Open another terminal session to get into AWS
- Go to your module5 directory and create a place to put the material we will download

```
cd ~/workspace/module5
mkdir great
cd great
```

- For you own analyses, you can download a bunch of tracks at the same time by using wget on a list of URLs
  - Use the **wget** command to download the text file that contains the list of tracks

```
wget -O trackList.txt 'https://epigenomesportal.ca/api/datahub/download?session=18731&format=text
```

- Now download the tracks that are contained in this list

```
wget -i trackList.txt
```

- Convert the bigbed using the UCSC set of tools

```
bigBedToBed 58394.Blueprint.ERS1027405.H3K27ac.peak_calls.bigBed 58394.Blueprint.ERS1027405.H3K27
```

If you're under Linux / Mac, you can also install the UCSC tools locally, as they are a useful set of tools to manipulate tracks data, without requiring so much processing power.

- GREAT has a limit on the number of regions to be tested in one execution. Therefore, we need to subsample our file. We can create a BED file subsample this way:
  - Sort BED file in random order with `sort -R`
  - Take the 20000 first lines in the file with head `-n20000`

```
sort -R 58394.Blueprint.ERS1027405.H3K27ac.peak_calls.bed > 58394.Blueprint.ERS1027405
head -n 20000 58394.Blueprint.ERS1027405.H3K27ac.peak_calls.random.bed > 58394.Bluepri
```

- From your local computer, download the BED file `58394.Blueprint.ERS1027405.H3K27ac.peak_c` locally using your browser

```
http://<your VM ip address>/module5/great/58394.Blueprint.ERS1027405.H3K27ac.peak_calls
```

- Load the GREAT website: http://bejerano.stanford.edu/great/public/html/
- Provide the following input to the GREAT interface:
  - Assembly: Human: GRCh38
  - Test regions: The randomized short version of the BED files you just downloaded (58394.Blueprint.ERS1027405.H3K27ac.peak_calls.random_short.bed)
  - Leave the "Background regions" to its default value, "Whole Genome"
- Submit the form
- In the results, for instance, you should obtain something like this for biological processes:

GO Biological Process (no terms) — Global controls

Table controls: Export — Shown top rows in this table: 20 Set — Term annotation count: Min: 1 Max: Inf Set — Visualize this table: [select one]

Term Name	Binom Rank	Binom Raw P-Value	Binom FDR Q-Val	Binom Fold Enrichment	Binom Observed Region Hits	Binom Region Set Coverage	Hyper Rank	Hyper FDR Q-Val	Hyper Fold Enrichment	Hyper Observed Gene Hits	Hyper Total Genes	Hyper Gene Set Coverage
myeloid leukocyte activation	6	2.3562e-169	5.1675e-166	2.1338	1,611	8.06%	60	5.0340e-17	1.3267	412	561	3.96%
myeloid cell activation involved in immune response	7	6.6326e-161	1.2468e-157	2.2003	1,435	7.18%	46	2.5491e-18	1.3558	382	509	3.68%
leukocyte activation involved in immune response	8	1.1109e-158	1.8273e-155	2.0389	1,678	8.39%	53	1.1227e-17	1.3212	441	603	4.24%
myeloid leukocyte mediated immunity	9	1.4408e-157	2.1066e-154	2.1866	1,425	7.12%	40	6.1744e-19	1.3611	385	511	3.70%
leukocyte degranulation	11	1.8824e-156	2.2519e-153	2.1999	1,397	6.99%	55	1.9995e-17	1.3504	373	499	3.59%
granulocyte activation	12	2.8809e-156	3.1592e-153	2.2130	1,378	6.89%	58	3.8350e-17	1.3503	367	491	3.53%
neutrophil activation involved in immune response	13	1.8380e-155	1.8604e-152	2.2318	1,347	6.74%	57	3.3254e-17	1.3549	360	480	3.46%
neutrophil activation	14	1.0410e-154	9.7851e-152	2.2066	1,373	6.87%	64	8.8327e-17	1.3475	364	488	3.50%
cell activation involved in immune response	15	3.0608e-153	2.6852e-150	2.0088	1,681	8.40%	54	1.7475e-17	1.3184	443	607	4.26%
neutrophil mediated immunity	16	1.4482e-152	1.1911e-149	2.2047	1,357	6.78%	51	7.1116e-18	1.3577	369	491	3.55%
neutrophil degranulation	17	3.9347e-152	3.0457e-149	2.2229	1,330	6.65%	61	5.8293e-17	1.3530	358	478	3.44%
leukocyte mediated immunity	19	2.0276e-151	1.4043e-148	2.0240	1,632	8.16%	85	5.3265e-15	1.2920	442	618	4.25%
Fc receptor signaling pathway	61	4.1045e-66	8.8544e-64	2.0815	674	3.37%	154	5.7030e-10	1.4334	146	184	1.40%

Bonus question: Why is your result slightly different from the screenshot?

8.2.5 Going back to HOMER Results

- Is the job done? If it is completed, you can bring back HOMER results to your laptop for visualization. First we'll compress the results to a zip file

```
cd ~/workspace/module5/homer/
zip -r homer.zip output
```

- Next, with your web browser, download the zipped result set

```
http://<your VM ip address>/module5/homer/homer.zip
```

- Unzip the file, and open the de novo and known motifs HTML files in a browser for visualization. Do the identified motifs fit what we would expect?

Homer Known Motif Enrichment Results (output)

Homer *de novo* Motif Results
Gene Ontology Enrichment Results
Known Motif Enrichment Results (txt file)
Total Target Sequences = 52145, Total Background Sequences = 51927

Rank	Motif	Name	P-value	log P-pvalue	q-value (Benjamini)	# Ta Seq with
1		CTCF(Zf)/CD4+-CTCF-ChIP-Seq(Barski_et_al.)/Homer	1e-36332	-8.366e+04	0.0000	257
2		BORIS(Zf)/K562-CTCFL-ChIP-Seq(GSE32465)/Homer	1e-28972	-6.671e+04	0.0000	264
3		NeuroD1(bHLH)/Islet-NeuroD1-ChIP-Seq(GSE30298)/Homer	1e-581	-1.339e+03	0.0000	742
4		Zic3(Zf)/mES-Zic3-ChIP-Seq(GSE37889)/Homer	1e-544	-1.253e+03	0.0000	682
5		Unknown-ESC-element(?)/mES-Nanog-ChIP-Seq(GSE11724)/Homer	1e-522	-1.204e+03	0.0000	610
6		CTCF-SatelliteElement(Zf?)/CD4+-CTCF-ChIP-Seq(Barski_et_al.)/Homer	1e-470	-1.084e+03	0.0000	726
7		Tgif2(Homeobox)/mES-Tgif2-ChIP-Seq(GSE55404)/Homer	1e-434	-9.999e+02	0.0000	198

8.2.5.1 All done!

If you have time remaining, you can explore further the tools that we covered in this lab, using other types of datasets. For example, does running a GREAT query on another cell type yield the type of annotations that you'd expect?

Interested in exploring Galaxy? You can start tackling a Galaxy introduction extra lab, available here. It uses the usegalaxy.org server, so you can also follow it at your own pace after the workshop.

Congratulations! You have completed Lab 5!