Lab 5: The dark matter of the genome (ChIP-seq)

Skills: ChIP-seq, genome browsers, peak calling, motif finding

For this week you'll need to complete the following:

- `CSE185-LAB5-EXERCISES1.ipynb` (10 pts)
- `CSE185-LAB5-EXERCISES2.ipynb` (10 pts)
- `CSE185-LAB5-REPORT.ipynb` (70 pts)
- `CSE185-LAB5-README.ipynb` (10 pts)

Similarly to the previous lab, you will complete your report in `CSE185-LAB5-REPORT.ipynb` and should document any code you used to complete the lab in `CSE185-LAB5-README.ipynb`.

Note: in this lab, we'll be generating some fairly large intermediate files (e.g. alignments). We have increased all student quotas to 25GB. To ensure you have enough space, please consider going through previous lab assignments that have already been graded and deleting large intermediate files you generated. Also see this post on disk quotas.

Intro

In 2006, there was a striking discovery that if you treat adult cells with a specific set of transcription factors, they could become "pluripotent", meaning they can then be programmed to theoretically any type of cell. These four transcription factors (Oct4/*Pou5f1*, Sox2, cMyc, and Klf4) are collectively known as "Yamanaka factors" (named after Shinya Yamanaka's lab, which originally showed this).

In this lab, we will analyze ChIP-sequencing from three of these factors (Oct4, Sox2, and Klf4) as well as two histone modifications (H3K4me2 and H3K27ac, which often mark regulatory/enhancer regions) in mouse embryonic stem cells, which are pluripotent and express the Yamanaka factors. We'll use the ChIP-seq data to determine where these factors and modifications are binding and which specific sequence motifs the factors are binding to.

In this lab, we'll go through:

1. Aligning ChIP-seq reads to a reference genome
2. Visualizing ChIP-seq data in IGV.
3. Identifying binding sites from ChIP-seq data ("peak calling")
4. Visualizing patterns of transcription factor and histone modification binding.
5. Motif finding to identify the sequence a transcription factor binds to.

Summary of tools covered

In this lab we'll be using the following tools:

- BWA MEM: for aligning our reads to the reference genome
- IGV: a genome browser. Used for visualizing ChIP-seq alignments and peaks (and other types of genomic data)
- HOMER: a toolkit that supports a large number of analyses of ChIP-seq datasets. These include:
  - Peak calling (identifying where the binding sites are from a ChIP-seq dataset)
  - Annotating peaks and generating coverage histogram plots
  - Motif finding (finding which sequences our transcription factor is binding to).

(Note, both IGV and HOMER are maintained by groups here at UCSD!)

And as usual, we'll do some plotting examples with the matplotlib Python library, but you can use whatever you're most comfortable with to plot.

Summary of data provided

Data for this week can be found in `~/public/lab5`. You should see:

- `Oct4.esc.fastq`: reads from ChIP-seq of the transcription factor Oct4.
- `Klf4.esc.fastq`: reads from ChIP-seq of the transcription factor Klf4.
- `Sox2.esc.fastq`: reads from ChIP-seq of the transcription factor Sox2.
- `H3K27ac.esc.fastq`: reads from ChIP-seq of the histone modification H3K27ac.
- `H3K4me2.esc.fastq`: reads from ChIP-seq of the histone modification H3K4me2.
- `input.esc.fastq`: reads from an input control (whole cell extract) for the ChIP-seq. Recall this comes from performing ChIP-seq but without the antibody pull down step.

For the motif analysis in the last part, you'll also see:

- `GRCm38.chr17.fa`: chr17 extracted from the GRCm38 reference genome.
- `OCT4_motif.meme`: the MEME-format motif for Oct4.

The mouse reference genome can be found at: `~/public/genomes/GRCm38.fa` (build GRCm38). For this week, we have also added the corresponding bwa index files in the same directory.

Acknowledgements: Parts of this lab are modified from material originally written by Chris Benner and revised by Alon Goren.

Tips

You may include an image file (such as an IGV screenshot) by:

- saving the image on your local computer (png files should work)
- navigating in JupyterHub to the assignment directory.
- uploading the file
- Using HTML (e.g. `<img src=filename>`) to include the file in a markdown cell

Be aware of chromosome name mismatches. e.g. "chr17" vs. "17".

For long running commands, `nohup` is your friend.

1. Aligning ChIP-seq reads

Before getting started, it is always a good idea to get some basic stats on the data you're dealing with.

Question 1 (5 pts) Summarize the datasets we are starting with: what samples are we analyzing? Where did they come from? How many reads are there in each dataset, and what is the read length? You can get this information by inspecting the fastq files provided and based on the description above.

The samples we are analyzing are the three transcription factors Oct4, Sox2, and Klf4 along with two histone modifications H3K4me2 and H3K27ac which are found in mouse embryonic stem cells. Additionally, there is an input control for the ChIP-seq. For the Oct4.esc.fastq file, there are 2539032 reads with a read length of 50. For the Klf4.esc.fastq file, there are 2750711 reads with a read length of 50. For the Sox2.esc.fastq file, there are 2634261 reads with a read length of 50. For the H3k27ac.esc.fastq file, there are 2362701 reads with a read length of 50. For the H3K4me2.esc.fastq file, there are 3126338 reads with a read length of 50. For the input.esc.fastq file, there are 2548616 reads with a read length of 50.

As in our previous NGS analyses, the first step will be to align the reads to a reference genome. We recommend using `bwa mem`. Type the command at the terminal to see usage, or look back to Lab 1 to recall the syntax for running BWA. The genome at `~/public/genomes/GRCm38.fa` has already been indexed using `bwa index`.

Align reads from each dataset to the GRCm38 (mm10) reference genome. You may want to use UNIX for loops to avoid retyping the command many times. You may also want to use multithreading (e.g. `-t 6`) to speed this up. We recommend using the high memory instances.

Store the results in `~/lab5`. After aligning, use `samtools` to sort and index the resulting BAM file (again, look back to Lab 1 if you need a refresher how to do this).

After alignment finishes, it is a good practice to take a look at the resulting BAM files using `samtools view`. e.g.:

```
samtools view mysample.bam | less -S
```

You can also see how many reads aligned to each chromosome using e.g.:

```
samtools view ~/lab5/bams/input.sorted.bam | cut -f 3 | uniq -c
```

(You should see that most reads were aligned to chr17, since we extracted only those reads to make this analysis run faster. But there will be reads aligned to other chromosomes as well which are mostly mapping errors).

Question 2 (5 pts) Summarize the methods you used to align the data. What aligner and version did you use? What build of the reference genome?

Hint: if you type `bwa` at the terminal it will print out a lot of information including the version.

To align the data, I used bwa-mem (alignment using Burrow-Wheeler transformation) which is using Version 0.7.17-r1198-dirty. I first indexed the GRCm38.fa reference genome using bwa index then used bwa-mem to align the samples to the reference genome which was saved as a sam file. I then used samtools (Tools for alignment in the SAM format) which was Version 1.9 (using htslib 1.9) to convert to a bam file, then used samtools sort to sort the newly created bam file and lastly used samtools index to index the newly sorted bam file. Then repeated this using each sample.

Question 3 (5 pts) What percentage of your reads from each dataset were successfully aligned? Recall `samtools flagstat` can be used to easily find this number.

(Alignment might take a couple minutes. In the mean time, you might want to get started installing and exploring IGV in part 3).

Oct4: 99.87% Sox2: 99.63% Klf4: 99.61% H3K4me2: 99.62% H3K27ac: 99.72% Input: 99.54%

2. Getting started with HOMER (making tag directories)

We will be using Homer for most of the analyses in this lab. For most analyses, Homer requires us to perform a preprocessing step to convert the BAM files into "tag directories". Tag directories are analogous to sorted BAM files and are the starting point for most HOMER operations like finding peaks, creating visualization files, or calculating read densities. The command also performs several quality control and parameter estimation calculations.

The command `makeTagDirectory` will do this for us. The syntax is:

```
makeTagDirectory <output directory> <input BAM file> [options]
```

For example, if you made a BAM file `~/lab5/bams/Oct4.sorted.bam`, you might run:

```
makeTagDirectory ~/lab5/tagdirs/Oct4 ~/lab5/bams/Oct4.sorted.bam
```

This command is going through the BAM file and doing lots of preprocessing steps: removing reads that do not align to a unique position in the genome, separating reads by chromosome and sorting them by position, calculating how often reads appear in the same position to estimate the clonality (i.e. PCR duplication), calculating the relative distribution of reads relative to one another to estimate the ChIP-fragment length, calculating sequence properties and GC-content of the reads and performing a simple enrichment calculation to check if the experiment looks like a ChIP-seq experiment (vs. an RNA-seq experiment).

The command creates a new directory, in this case named `~/lab5/tagdirs/Oct4`. Inside the directory are several text files that contain various QC results. Don't worry too much about those, but you can try looking at the output files by typing `less -S <filename>` to see what's there.

Run this on all of your aligned BAM files. It shouldn't take long (seconds) for each sample to finish. Be sure to document the commands you used to do this in your README file.

3. Visualizing the data with IGV

Next we will visualize the ChIP-seq experiments by creating bedGraph files from the tag directories and using the IGV genome browser to look at the results. BedGraph files are similar to BED files we've seen in the past (with chrom, start, and end columns) except now with a 4th column giving a datavalue. These files are useful for describing how many reads aligned to each region of the genome.

We will create these files using the `makeUCSCfile` command that is part of Homer. For most ChIP-seq experiments all you need to do is specify the tag directory and specify `-o auto` for the command to automatically save the bedGraph file inside the tag directory. e.g.:

```
makeUCSCfile ~/lab5/tagdirs/Oct4 -o auto
```

This creates the file `~/lab5/tagdirs/Oct4/Oct4.ucsc.bedGraph.gz`. This file format specifies the normalized read depth at variable intervals along the genome (use `zcat` and the filename to view the file format for yourself).

Use the `makeUCSCfile` command to make bedGraph files for each of your samples.

Now, to visualize with IGV, first download the bedGraph files to your local computer. You can download files from datahub by navigating to the directory where the file is stored, clicking the box next to it, and clicking "download".

Then, go to IGV. There is a desktop version and a javascript version (https://igv.org/app/). We recommend the desktop version, which can be easier to use and faster. Instructions below are the mostly the same regardless of which one you use.

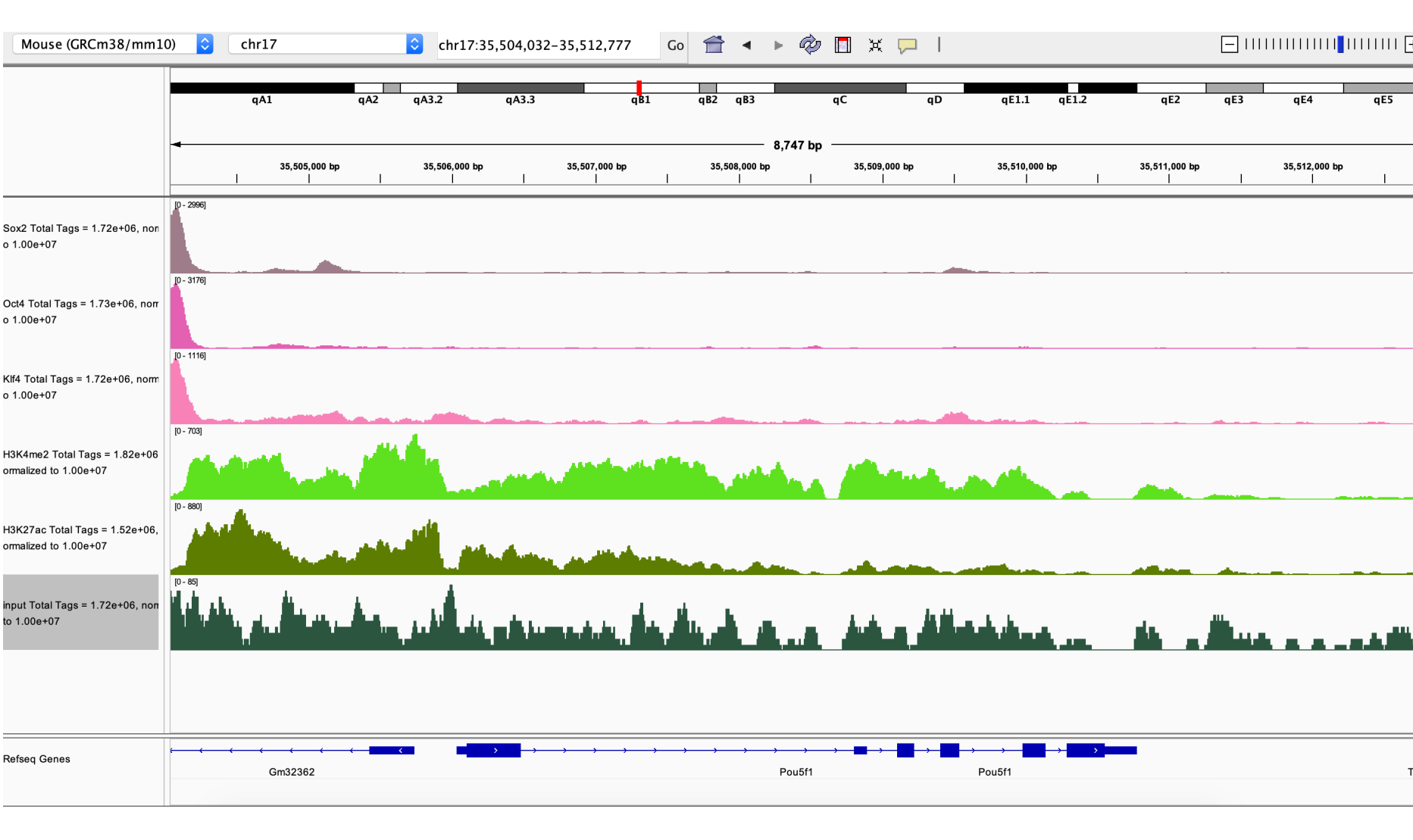
Make sure to select the "mm10" genome (which is equivalent to GRCm38). Use "Tracks->Local file" to upload your bedgraph files.

The tracks will display the relative density (coverage) of ChIP-seq reads at each position in the genome. Zoom in chr17, since that is where all of data is from.

See if there are any interesting patterns in the data that catch your eye. Try visiting the *Pou5f1* locus (the gene for Oct4) by typing the gene name into the search bar at the top. Once at the *Pou5f1* locus, zoom out to see if there any nearby sites with possible binding sites in each of your tracks.

Question 4 (10 pts) Based on visual inspection of the datasets on IGV, where are the peaks for each of the datasets falling (promoters? gene regions? exons? elsewhere?). How do the signals for the transcripton factors (Oct4, Sox2, and Klf4) compare to the signals for the two histone modifications? What does the input signal look like? Include a screenshot of the IGV visualization in your report. The lecture slides provide some hints for the types of patterns you should be seeing for each dataset.

Based on the visual inspection of the datasets on IGV, the peaks for Oct4, Sox2, and Klf4 fall primarily in the promoter region and are very "focal" and definitive. Compared to the peaks of the transcription factors, the histone modifications are more spread out throughout the gene regions and are more "broad". Input's peaks are more staggered-like and are all through the gene regions as well but more than the transcription factors and histone modifications.



4. Identifying binding sites (peaks)

One of the most common tasks with ChIP-seq data is to find regions that enriched (compared to a control like whole cell extract). These enriched regions are commonly called "peaks". We will use the `findPeaks` utility from Homer, which takes tag directories as input and outputs a set of peak regions.

The general syntax of the command is:

```
findPeaks <tag directory> -i <control tag directory> -style XX -o auto
```

for example:

```
findPeaks ~/lab5/tagdirs/Oct4 -i ~/lab5/tagdirs/input -style factor -o auto
```

where we will use `-style factor` for transcription factors and `style histone` for histone modifications. Generally, for transcription factors we will be looking for more focal, narrower peaks whereas for histone modifications we will be looking for broader peaks. This flag tells Homer which type of peaks to look for.

This command will look for enriched regions and filter them based on several criteria, including ensuring that they have at least 4-fold more reads in peak regions relative to the control experiment (in this case `input`). The output will be stored in a HOMER-style peak file located in the Oct4 tag directory (`Oct4/peaks.txt`). This file will be called `regions.txt`, rather than `peaks.txt`, for histone modifications. The beginning of this file contains statistics and QC stats from the peak finding, including the number of peaks, number of peaks lost to input filtering, etc.

One field worth paying attention to is the "Approximate IP efficiency" which reports what fraction of reads from the experiment were actually found in peaks. For most decent experiments this value ranges from 1% to >30% (remember ChIP is an enrichment strategy... there is plenty of background in the data too!).

After the header lines (which begin with `#`), you'll see the actual peaks, with one line per peak. There are a lot of columns. The important ones for us are: columns 2-4 give the chromosome start end, column 11 gives the fold change over the background, and column 12 gives the p-value.

Use `findPeaks` to call peaks in each sample. Be sure to set `style` appropriately for each dataset.

Question 5 (8 pts) Summarize the methods you used to call peaks for each sample the data. Then, summarize peak-finding results: what was the IP efficiency reported for each dataset? How many peaks did you find? What was the average peak size for each dataset? Note you should only have 5 peak sets, since you won't have peaks for the input control.

The method I used was similar to the syntax above where I called findPeaks on the input control directory and used it on each sample's directory with style being "factor" for the TF and style being "histone" for the histone modifications. I then just iterated through each sample's tag directory and called each directory once for a total of 5 sets. Then, I checked the returned peaks.txt or regions.txt file to summarize peak-finding results.

Oct4 - IP Efficiency: 2.65% Peaks Found: 2203 Average Peak Size: 75

Klf4 - IP Efficiency: 3.12% Peaks Found: 3020 Average Peak Size: 75

Sox2 - IP Efficiency: 2.74% Peaks Found: 2236 Average Peak Size: 75

H3K27ac - IP Efficiency: 28.57% Peaks Found: 2102 Average Peak Size: 500

H3K4me2 - IP Efficiency: 70.02% Peaks Found: 3060 Average Peak Size: 500

One other thing to note is that HOMER reports the results in a "peak" file, which has a slightly different format from a traditional BED file format. To create a BED file from the peak file, use the Homer utility `pos2bed.pl` following the sample below. BED files can be uploaded to IGV just like a bedGraph file. Also, most HOMER programs will work with either BED or peak files as input. Example command:

```
pos2bed.pl Oct4/peaks.txt > Oct4/Oct4.peaks.bed
```

Copy the peaks (in BED format) you called to your local computer, and upload the resulting files to IGV. Explore how they look on IGV: what does the coverage profile look like for peaks with high scores? low scores?

5. Visualizing binding patterns

We will next use the Homer utility `annotatePeaks.pl` to learn about where our peaks are falling. This tool can do a lot of different things. For now we'll use it to visualize read counts around promoter regions.

The general usage of `annotatePeaks.pl` is:

```
annotatePeaks.pl <peaks file> <genome fasta> [options] > output.txt
```

We can use the `-hist` option to `annotatePeaks.pl`, which gives the relative count of reads for each dataset centered around a specific feature. Here, we will look at read counts relative to transcription start sites (TSSs), which mark the center of promoter regions. The command below can be used to create histogram data for a list of tag directories. Note you'll have to change this command if your tag directories had different names.

```
annotatePeaks.pl tss \
  ~/public/genomes/GRCm38.fa \
  -size 8000 \
  -hist 10 \
  -d ~/lab5/tagdirs/Oct4 ~/lab5/tagdirs/Sox2 ~/lab5/tagdirs/Klf4 ~/lab5/tagdirs/H3K4me2 ~/lab5/tagdirs/H3K27ac \
  -gtf ~/public/genomes/GRCm38.75.gtf > ~/lab5/annotations/tss_histogram.txt
```

Here is a brief description of the options used:

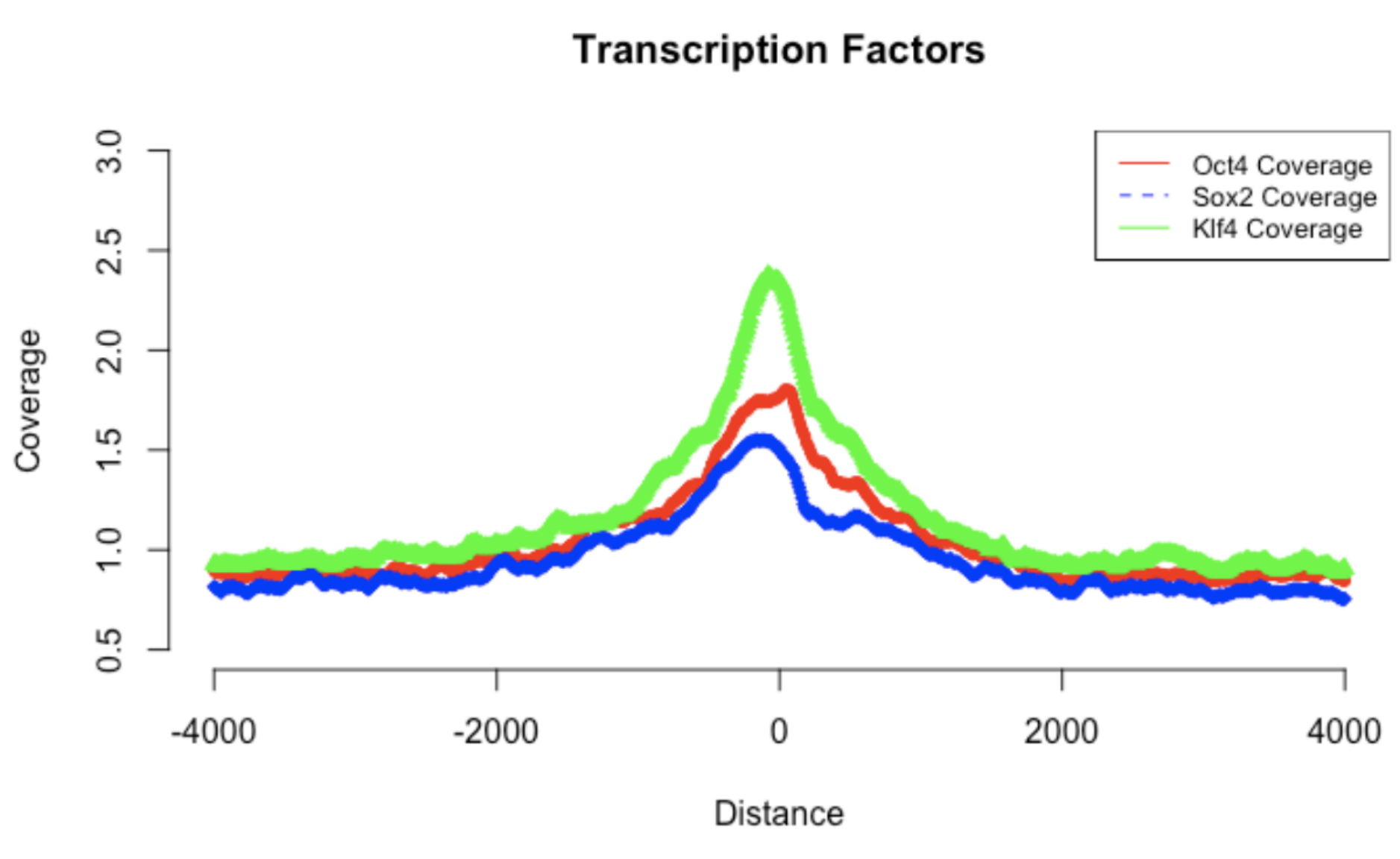
- `tss` is a built-in shortcut telling Homer to look at regions centered on transcription start sites. We could have also provided a peaks file with regions to plot or a BED file listing locations of transcription start sites.
- `GRCm38.fa` tells it where the reference genome is
- `-size 8000` tells it look at 8000bp regions centered at each TSS
- `-hist 10` gives the bin size for our histograms
- `-d tagdir1 tagdir2...` gives a list of tag directories
- `-gtf` points it to the gene annotations, which it needs to figure out where the TSSs are.

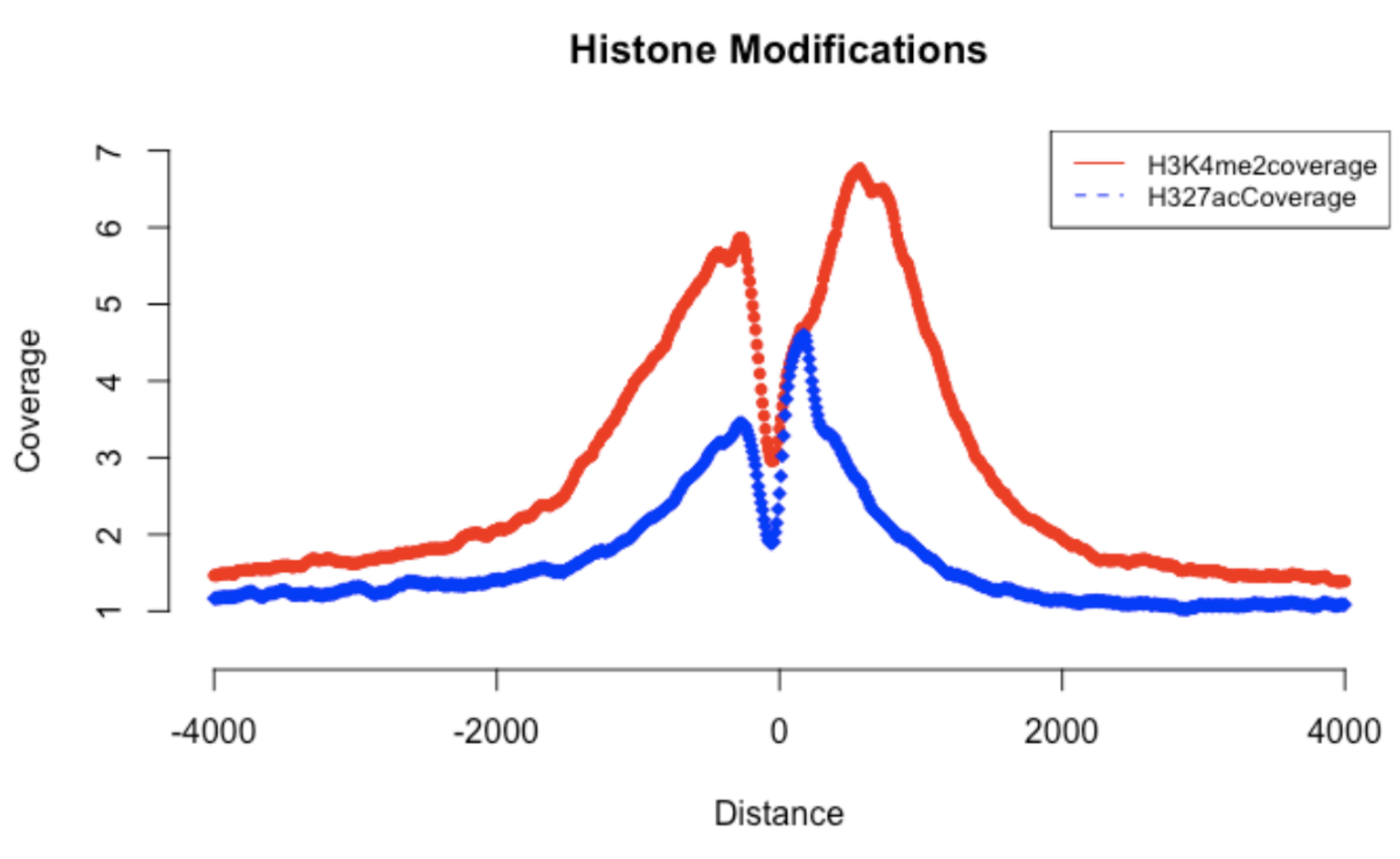
Take a look at the output file. You'll notice that the first column gives the distance offsets from the TSS followed by columns corresponding to the 'coverage', '+ Tags', and '- Tags' for each experiment.

Question 6 (10 pts) Generate a composite plot showing the distribution of read counts each of the datasets around TSSs. Plot transcription factors and histone modifications on separate plots to more easily see differences. Your plot should show:

- x-axis: gives the distance from the TSS. It should range from -4000 (4kb upstream of TSSs) to +4000 (4kb downstream).
- y-axis: gives the relative read (tag) count. This is given in the columns with "Coverage" in the header in the `tss_histogram.txt` file generateed above.
- You should have one line per dataset, with a separate color for each.
- Examples of what this should look like are given in the lecture slides.

Be sure to label your axes and provide a legend or description of which color denotes which dataset.





6. Motif finding

Now that we have an idea where these factors are binding, we'd like to know what type of sequences they bind to. Motif finding is a powerful technique to identify specific patterns of DNA sequence that are bound by a particular transcription factor. To use HOMER's motif analysis program, run the findMotifsGenome.pl command using peak files from the experiments. In general the command has syntax:

```
findMotifsGenome.pl <peaks> <refgenome> <output directory> [options]
```

For example:

```
prefix=Oct4
findMotifsGenome.pl \
 ~/lab5/tagdirs/${prefix}/peaks.txt \
 ~/public/genomes/GRCm38.fa \
 ~/lab5/motifs/${prefix} \
 -mask -size 100
```

This will find motifs enriched in Oct4 peaks, and output the results to the directory `~/lab5/motifs/Oct4`. The options at the end tell it to mask repeats when performing the analysis, and to look at regions of size 100 centered at each peak region.

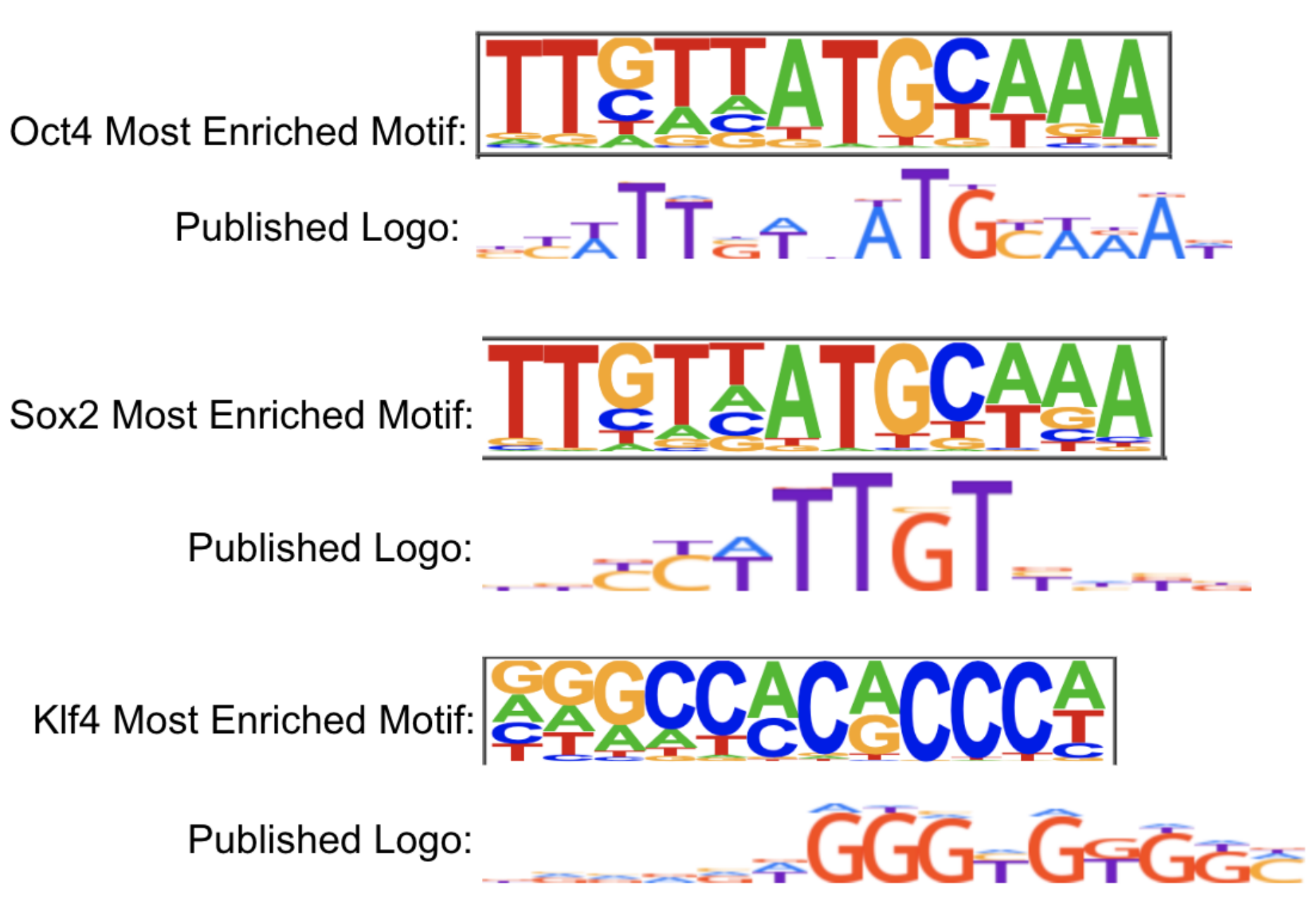
This will create, among other things, a file `~/lab5/motifs/Oct4/homerResults.html` containing motifs that Homer found to be significantly enriched in our set of peaks. Navigate through Jupyter to view this html file in your browser.

Run motif finding for the three transcription factors (don't do this for the histone modifications, which aren't likely to give good motifs since they do not recognize specific sequences like transcription factors do. Of course if you really want to try it go ahead). This can take a while. You can speed it up with e.g. `-p 5` to use multithreading. Or use `nohup` to run it in the background and come back to it later.

Question 7 (7 pts) Briefly summarize the methods you used to perform motif finding. Which motifs did you find to be most enriched for each factor? How do these compare to published logos for these factors? Include a figure comparing them.

See: http://hocomoco11.autosome.ru/mouse/mono?full=false for published motifs for many mouse factors, including Klf4, Pou5f1/Oct4, and Sox2.

To perform motif finding I used the above code which will first call findMotifsGenome.pl which will find the most enriched motifs in Oct4 peaks by giving the peaks.txt file of the Oct4 tagdir in as the input directory. I also used -mask to mask repeats and used -size 100 to look at regions centered at each peak region of size 100. This outputted the homerResults.html file to the Oct4 motifs directory which I then found the most enrinched motif and compared it to the published logos. I then repeated this two more times but switched out Oct4 for Sox2 then switched out Sox2 for Klf4. I found the most enriched motif for Oct4 to be TTGTTATGCAAA, this was compared to the published logo of Pou5f1/Oct4 which is CCATTGT_ATGCAAAT and shares the same "TTGTATGCAAA" sequence. I found the most enrinched motif for Sox2 to be TTGTTATGCAAA, this was compared to the published logo of Sox2 which is TTCCTTTGTTTTG and only shares the same "TTG" sequence. I found the most enriched motif for Klf4 to be GGGCCACACCCA, this was compared to the published logo of Klf4 which is TGGAGTGGGTGTGGC and doesn't really share a sequence. Below is an image showing how I aligned them up to compare:



7. Differential binding

You might have noticed a strong similarity between the motifs of Sox2 and Oct4! On IGV you'll see these often bind together. In this section, we'll see if we can find a motif that is specific to Sox2 binding, rather than Sox2+Oct4 binding.

First, merge the peak sets from the two factors so you just have one set of regions to analyze. You can use the Homer `mergePeaks` command for this. e.g.:

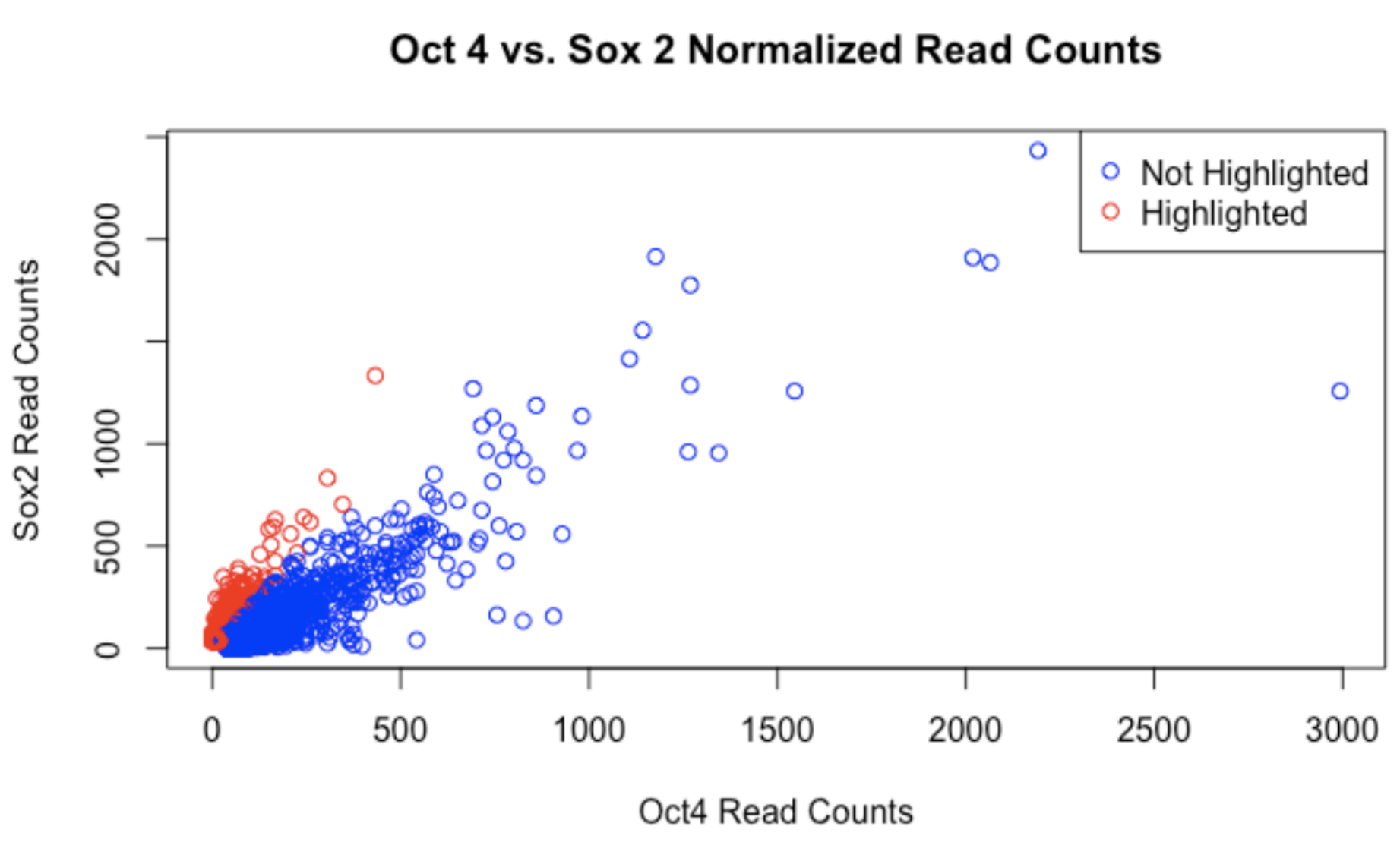
```
mergePeaks ~/lab5/tagdirs/Oct4/peaks.txt ~/lab5/tagdirs/Sox2/peaks.txt >
~/lab5/overlap/oct4_sox2_peaks_merged.txt
```

Now, you can compare read counts from Oct4 vs. Sox2 in these merged regions. The `annotatePeaks.pl` tool can help us get these read counts:. e.g.:

```
annotatePeaks.pl \
   ~/lab5/overlap/oct4_sox2_peaks_merged.txt \
  ~/public/genomes/GRCm38.fa \
  -d ~/lab5/tagdirs/Oct4 ~/lab5/tagdirs/Sox2 > ~/lab5/overlap/oct4_sox2_scatter.txt
```

Take a look at the output file (`oct4_sox2_scatter.txt`) above. The last two columns should have normalized tag counts for Oct4 and Sox2.

Question 8 (6 pts) Make a scatter plot comparing normalized read counts for Oct4 vs. Sox2 that you got from the merged peak set. Highlight peaks (using a different color) that correspond to peaks with two-fold higher binding in Sox2 vs. Oct4.



Question 9 (6 pts) Repeat motif finding, but this time using only the peaks that were unique to Sox2 (fold change >2). Report the motif you found. Is this different than the motif you found when analyzing *all* Sox2 peaks? Does this match to published motifs for Sox2?

The motif I found was TTCCTTTGTTTTG and it was different than TTGTTATGCAAA, the motif I found when analyzing all Sox2 peaks. This motif does match to the published motifs for Sox2 which is also TTCCTTTGTTTTG for the SOX2_MOUSE.H11MO.1.A Model.

Discussion Questions

Question 10 (4 pts) You should find that two of the transcription factors have very similar motifs to each other. Which two? Read about these factors (e.g. wikipedia) and hypothesize why we found these factors binding to the same motif.

The two transcription factors that have very similar motifs to each other are Oct4 and Sox2 which have the same most enriched motif of TTGTTATGCAAA. We would find these factors binding to the same motif because they both have crucial roles in embryonic stem cell development and pluripotency. They are also known to cooperate together and often bind to the same or overlapping DNA motifs in regulatory regions of target genes. Thus, I believe that these two factors binded to the same motif because I think that TTGTTATGCAAA is in the regulatory region in order to regulate gene expression as Oct4 and Sox2 work together to contibute to the same processes of governing embryonic development.

Question 11 (4 pts) There are many Oct4 motifs in the genome that are not actually bound by the Oct4 TF. Why do you think that is? Besides motif occurrence, what else do you think we could use to determine whether a TF is bound to a particular motif? (There are many possible answers).

One reason I think that is is similar to my previous answer about cooperative binding. I think that Oct4 may require other TF to bind to the same motif so maybe since some motifs didn't have the other TF bound to it then the Oct4 TF may not have been able to bind alone. In order to determine whether a TF is bound to a particular motif or not, we could also use ChIP-seq which indentigies regions in the genome where a TF is bound.

References

To read more about the data used for this lab see: https://www.ncbi.nlm.nih.gov/pubmed/28111071