Analysis of ATAC-seq data

Michael J. Guertin

June 29, 2019

A step-by-step procedure for ATAC-seq analysis [1]

Contents

1 Processing ATAC-seq reads **3**
1.1 Concatenating FASTQ files . . . 3
1.2 Aligning to Genome . . . 4
1.2.1 index genome . . . 4
1.2.2 Align *fastq.gz* files with bowtie2 . . . 4
1.3 Convert to *bam* file format . . . 5
1.4 `seqOutBias` gets its own subsection because, like enzymes, I am biased . . . 5
1.5 Merging processed BAM files for visualization and non-IDR peak calling methods, such as `hotspot` . . . 5
1.6 Converting merged BAM files to bedGraph files for uploading directly to UCSC . . . 6

2 Peak calling with `hotspot` **6**
2.1 Generation of `hotspot` input files . . . 7

3 Peak calling with `macs2` **8**
3.1 Running `macs2` . . . 8

4 Analysis in R **9**
4.1 Principle Components Analysis of replicates and conditions . . . 9
4.2 Differential Accessibility + 1hr. DMSO vs. ScriptAid . . . 12

5 Motif Analysis **16**
5.1 MEME . . . 16
5.2 FIMO . . . 23

6 Polishing in Adobe Illustrator **29**

7 Exploratory project **30**
7.1 Comparison of peak callers . . . 31
7.2 Comparison of MEME 4.12 to MEME 5.0 . . . 32

List of Figures

1 UCSC browser decreased peak . . . 7
2 UCSC browser increased peak . . . 7
3 Hotspot PCA . . . 11
4 MACS2 PCA . . . 11
5 Original Hotspot DESeq2 . . . 14
6 Pooled MACS2 1 hour DESeq2 . . . 15
7 Pooled MACS2 7 day DESeq2 . . . 15
8 motif seqLogos enriched in peaks that decrease intensity . . . 17

[1] address questions to Michael Guertin, guertin@virginia.edu

9 motif seqLogos enriched in peaks that increase intensity 18
10 motif seqLogos enriched in MACS2-called peaks that decrease intensity at 1 hour 19
11 motif seqLogos enriched in MACS2-called peaks that increase intensity at 1 hour 20
12 motif seqLogos enriched in MACS2-called peaks that decrease intensity at 7 days 21
13 motif seqLogos enriched in MACS2-called peaks that increase intensity at 7 days 22
14 Instances of motifs relative to ATAC peak summits 26
15 Instances of motifs relative to MACS2-called ATAC peak summits at 1 hour 26
16 Instances of motifs relative to MACS2-called ATAC peak summits at 7 days 27
17 The TEAD family motif is present in a significantly higher proportion of peaks that increased accessibility as compared with the other classes of peaks. 28
18 Polished Figure 29

1 Processing ATAC-seq reads

The files came in folders labeled like this: U412_7dayDMSO3_L004-ds.057b4de30825493d938cc3fa28094c38 I don't know what the U412 is, but I suspect that it has something to do with the machine or the run number.
Fortunately, these samples are labeled descriptively, so we can combine everything by the descriptive labels. This part is foolproof because the same barcodes are applied to each lane. The L001-L004 is a lane number. NextSeq has 4 lanes, but the same library has to go in each lane, not like other platforms where different libraries go in different lanes. But our libaries are multiplexed (barcoded), so we don't waste sequencing. That last part is obviously garbage text and this whole data labeling scheme is annoying because our core used to send us the data in a more reasonable format.
Within these directories we have files labeled: 1574-7dayDMSO3_S3_L004_R1_001.fastq.gz I don't know what the first number is, but the description is preserved within the folder. I assume that the S3 is the index (bar code). Check with Arun to confirm this. If this is the case, please confirm that the barcodes are associated with the correct decriptions. In the early days I had an issue with the Core at Cornell mislabeling a sample. We will perform a PCA analysis with the replicates that will detect this sort of issue, but it will only cluster replicates, so it is possible that all replicates could be mislabeled, but then the biology would have to inform on any issues. R1 and R2 are paired files from paired end reads. The last 001 refers to the file number–they like to keep individual files small, so there are often many of these if we don't barcode too many samples.

1.1 Concatenating FASTQ files

All the files should be named with the description, the replicate number, and then the paired end number. No spaces. This makes it so that downstream analyses can be automated. I would recommend in the future that one labels the cell type, these are cancer associated fibroblasts from Dr. Drew Dudley's mice. We need to relabel them anyways when we concatenate files. This is probably the most important code to be careful with, because your names will carry through to all downstream analyses and automatically name output files. Once I cavalierly used a wildcard (*) and all my files labeled samples 11-19 ended up in the 1 sample because I put a wildcard after my 1–files 2-9 were processed fine, so no red flags went up until later in my analysis.
Here I provided sample code for processing three replicates. This is the only hard coding you need to do, everything else is automated in shell scripts, but I will break the scripts down for you. I strictly run pipelines with shell script loops because I like the flexibility they provide. Dr. Nathan Sheffield in CPHG likes more well annotated pipelines with version control in github, but this can be a very involved process. Nathan and I are two extremes, but as long as your analyses are reproducible and annotated, find what works for you.

```
cat *7dayDMSO1*/*_R1_*.fastq.gz > CAF_DMSO_7d_rep1_PE1.fastq.gz
cat *7dayDMSO1*/*_R2_*.fastq.gz > CAF_DMSO_7d_rep1_PE2.fastq.gz
cat *7dayDMSO2*/*_R1_*.fastq.gz > CAF_DMSO_7d_rep2_PE1.fastq.gz
cat *7dayDMSO2*/*_R2_*.fastq.gz > CAF_DMSO_7d_rep2_PE2.fastq.gz
cat *7dayDMSO3*/*_R1_*.fastq.gz > CAF_DMSO_7d_rep3_PE1.fastq.gz
cat *7dayDMSO3*/*_R2_*.fastq.gz > CAF_DMSO_7d_rep3_PE2.fastq.gz

cat *7dayScriptaid1*/*_R1_*.fastq.gz > CAF_Scriptaid_7d_rep1_PE1.fastq.gz
cat *7dayScriptaid1*/*_R2_*.fastq.gz > CAF_Scriptaid_7d_rep1_PE2.fastq.gz
cat *7dayScriptaid2*/*_R1_*.fastq.gz > CAF_Scriptaid_7d_rep2_PE1.fastq.gz
cat *7dayScriptaid2*/*_R2_*.fastq.gz > CAF_Scriptaid_7d_rep2_PE2.fastq.gz
cat *7dayScriptaid3*/*_R1_*.fastq.gz > CAF_Scriptaid_7d_rep3_PE1.fastq.gz
cat *7dayScriptaid3*/*_R2_*.fastq.gz > CAF_Scriptaid_7d_rep3_PE2.fastq.gz

cat *1hrDMSO1*/*_R1_*.fastq.gz > CAF_DMSO_1h_rep1_PE1.fastq.gz
cat *1hrDMSO1*/*_R2_*.fastq.gz > CAF_DMSO_1h_rep1_PE2.fastq.gz
cat *1hrDMSO2*/*_R1_*.fastq.gz > CAF_DMSO_1h_rep2_PE1.fastq.gz
cat *1hrDMSO2*/*_R2_*.fastq.gz > CAF_DMSO_1h_rep2_PE2.fastq.gz
cat *1hrDMSO3*/*_R1_*.fastq.gz > CAF_DMSO_1h_rep3_PE1.fastq.gz
cat *1hrDMSO3*/*_R2_*.fastq.gz > CAF_DMSO_1h_rep3_PE2.fastq.gz

cat *1hrScriptaid1*/*_R1_*.fastq.gz > CAF_Scriptaid_1h_rep1_PE1.fastq.gz
cat *1hrScriptaid1*/*_R2_*.fastq.gz > CAF_Scriptaid_1h_rep1_PE2.fastq.gz
cat *1hrScriptaid2*/*_R1_*.fastq.gz > CAF_Scriptaid_1h_rep2_PE1.fastq.gz
cat *1hrScriptaid2*/*_R2_*.fastq.gz > CAF_Scriptaid_1h_rep2_PE2.fastq.gz
cat *1hrScriptaid3*/*_R1_*.fastq.gz > CAF_Scriptaid_1h_rep3_PE1.fastq.gz
cat *1hrScriptaid3*/*_R2_*.fastq.gz > CAF_Scriptaid_1h_rep3_PE2.fastq.gz
```

1.2 Aligning to Genome

Retrieve the relevant genome from UCSC (Karolchik *et al.*, 2014), we will use the latest assembly, hg38. This is a zipped *fasta* file of the entire human genome. Bowtie 2 is an efficient tool for aligning sequencing reads to long reference sequences. For this execution we used Bowtie2 version 2.2.6. The first task is to build the genome index with bowtie2 (Langmead *et al.*, 2009) `http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#the-bowtie2-build-indexer`. This only has to be performed once per genome. Can you do this for mm10? You cannot get the *mm10.fa* file directly. You will need to run `twoBitToFa` on the *mm10.2bit* compressed file. I included the FASTA file in this directory, but you should try it and modify the chunk below for mm10.

1.2.1 index genome

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz
gunzip hg38.fa.gz
bowtie2-build hg38.fa hg38

wget http://hgdownload.soe.ucsc.edu/goldenPath/mm10/bigZips/mm10.2bit
twobittofa mm10.2bit mm10.fa
bowtie2-build mm10.fa mm10
```

1.2.2 Align *fastq.gz* files with bowtie2

Next we will use bowtie2 to align to the indexed mouse genome files. Bowtie2 is well documented, but the arguments we use are `-1` and `-2` for the paired end reads. What argument would you use for a single unpaired FASTQ file? The `-S` argument writes to a sam file, which is readable. However, the default is to write to `stdout`, which you will want to use to make a single line of code to go from raw FASTQ files to the final BAM without writing all the intermediate files, as I do here. The `-x` argument is the location of all your indexed genome files, they all have the file extension `.rev.*.bt2` and `.*.bt2`. Eventually I want to compare results that include all reads (default is concordant reads with less than 500bp intervening between the adapters) with results from only using subnucleosomal (<140bp) insertion events, argument `-maxins`. The original DNase-seq protocols selected these small species on sucrose gradients, but ATAC experiments and analyses include larger fragments. There are some reasons to include these fragments the original ATAC-seq paper goes over what these segments represent (Buenrostro *et al.*, 2013), but excluding larger fragments looks more like DNase-seq and represents DNA that is liberated in nucleosome depleted regions, where transcription factors are bound. Therefore, we are interested in these regions, but I suspect that Arun will do a side-by-side comparison to see if we get different results in the differential accessibility analyses and *de novo* motif analyses with these different populations of DNA fragment sizes. Running these commands prints statistics to the screen, so you can calculate the fraction of concordantly aligned reads with the 500bp default are capture when this is limited to 140bp.

```
bowtie2 -x /Users/guertinlab/Desktop/Dudley_Lab/mm10 -1 CAF_Scriptaid_1h_rep1_PE1.fastq.gz \
-2 CAF_Scriptaid_1h_rep1_PE2.fastq.gz -S CAF_Scriptaid_1h_rep1_500.sam

bowtie2 --maxins 140 -x /Users/guertinlab/Desktop/Dudley_Lab/mm10 -1 CAF_Scriptaid_1h_rep1_PE1.fastq.gz \
-2 CAF_Scriptaid_1h_rep1_PE2.fastq.gz -S CAF_Scriptaid_1h_rep1.sam
```

This code will loop through files in a directory. The file name is split on the '.fastq.gz' string and the variable 'name' is assigned to the first string.

```
for fq in *rep*_PE1.fastq.gz
do
    name=$(echo $fq | awk -F"_PE1.fastq.gz" '{print $1}')
    echo $name
    bowtie2 --maxins 140 -x /Users/guertinlab/Desktop/Dudley_Lab/mm10 -1 ${name}_PE1.fastq.gz \
-2 ${name}_PE2.fastq.gz -S ${name}.sam
done
```

1.3 Convert to *bam* file format

Sam files retain all the information from the *fastq* files, but include additional information, including alignment coordinates (http://samtools.github.io/hts-specs/SAMv1.pdf). The header has all the chromosome size information from the mm10.fa file.
We first convert the sam file to the compressed and **sorted** BAM format using `samtools` (Li *et al.*, 2009). Then we remove duplicate paired reads. Since we are not using unique molecular identifiers we are going to assume that reads that start in the same position on both ends are PCR amplicon duplicates. The `-s` argument (lowercase) specifies this.

```
for fq in *rep*_PE1.fastq.gz
do
    name=$(echo $fq | awk -F"_PE1.fastq.gz" '{print $1}')
    echo $name
    samtools view -b $name.sam -o $name.unsorted.bam
    samtools sort -n $name.unsorted.bam -o $name.name.bam
    samtools fixmate -m $name.name.bam $name.fixmate.bam
    samtools sort $name.fixmate.bam -o $name.sorted.fixmate.bam
    samtools markdup -r $name.sorted.fixmate.bam $name.rmdup.mark.bam
done
```

Aligning through BAM file processing can all be executed on a single line of code through the use of pipes. By using a sample FASTQ file with only a few hundred thousand reads (use a multiple of 4–`head` the FASTQ file to see the four line format), try to build your own pipe line. Google as needed.

The command below does everything in one step, but you may want to save intermediate files so we can easily calculate percent duplicated and other metrics, we used to calculate mitochondiral reads, but the OMNI protocol seems to limit mitochondiral incorporation. The digitonin compromises the nuclear membrane and Tn5 traverses more efficiently.

```
for fq in *rep*_PE1.fastq.gz
do
    name=$(echo $fq | awk -F"_PE1.fastq.gz" '{print $1}')
    echo $name
    samtools view -b -u $name.sam | samtools sort -n - | \
samtools fixmate -m - - | samtools sort - |  \
samtools markdup -r - $name.rmdup.mark.bam
    gzip $name.sam
done
```

1.4 `seqOutBias` gets its own subsection because, like enzymes, I am biased

We typically use the R `bigWig` library from André Martins (https://github.com/andrelmartins/bigWig) to operate on bigWig files directly in R for counting and downstream analyses. Here we are not implement enzyme bias scaling of the ATAC-seq reads but `seqOutBias` (Martins *et al.*, 2018) has some useful features (e.g. –shift-counts) that we will use to make `bigWig` files. The time consuming intermediate files are only made once, so only the first run is slow.

I am not going to go over the arguments, because a *wonderfully written* and *exceptionally complete* description of the methods, algorithms, software, and biological implications exist as a PDF vignette (https://guertinlab.github.io/seqOutBias/seqOutBias_vignette.pdf), a website vignette (https://guertinlab.github.io/seqOutBias_Vignette/), a user guide (https://guertinlab.github.io/seqOutBias/seqOutBias_user_guide.pdf), and a paper (Martins *et al.*, 2018).

```
for fq in *rep*_PE1.fastq.gz
do
    name=$(echo $fq | awk -F"_PE1.fastq.gz" '{print $1}')
    echo $name
    seqOutBias ~/Desktop/Dudley_Lab/mm10.fa $name.rmdup.mark.bam --skip-bed \
--no-scale --bw=${name}.bigWig --only-paired --shift-counts --read-size=38
done
```

1.5 Merging processed BAM files for visualization and non-IDR peak calling methods, such as `hotspot`

You want to merge BAM files for visualization on UCSC or your favorite browser and for peak callers that don't use irreproducible discovery rate methods involving replicates. As a first pass I would

recommend merging replicates for visualization, but merging all files for a cell type should be the most sensitive at finding peaks. Usually peaks change intensity between conditions (at least short treatments) and are not lost all together. Below, I hard coded the merging of three replicates, but this will change depending on replicate number. A shell script that writes all relevant BAM replicates to a file by using `touch`, `ls`, and other shell functions could be used in combination with the `-b` parameter of `samtools merge` to generalize this process. As opposed to hard coding the merging of all BAM files try to write a set of commands to merge all BAMs for peak calling. At this point we don't have good QC metrics for any of the libraries except by looking at the bowtie alignments. We will systematically QC libraries later, but for now I would just exclude merging any files with alignment metrics that are outliers. read depth is not a big issue, but percent aligned concordantly should be consistent, albeit lower than if we were not using the -minsize of 140.

```
for fq in *rep1_PE1.fastq.gz
do
    name=$(echo $fq | awk -F"_rep1_PE1.fastq.gz" '{print $1}')
    echo $name
    samtools merge ${name}_merged.rmdup.bam \
${name}_rep1.rmdup.mark.bam ${name}_rep2.rmdup.mark.bam ${name}_rep3.rmdup.mark.bam
done

#Creates a pan-merged file if you want to merge before calling peaks
samtools merge CAF_panmerged.rmdup.bam *_merged.rmdup.bam
```

1.6 Converting merged BAM files to bedGraph files for uploading directly to UCSC

This section needs some work, but it is functional as written. The biggest issue with this is that the alphabetically named chromosomes will not be included due to the limitation in file size. I suspect that trainees will find a browser that they like better than UCSC that they will use or we will have to start using UVA servers to host bigWigs files for UCSC access.

```
for bam in *_merged.rmdup.bam
do
    name=$(echo $bam | awk -F".rmdup.bam" '{print $1}')
    echo $name
    samtools view -h -o ${name}.sam $bam
    python /Users/guertinlab/pyscripts/samToBed.py -s ${name}.sam -t con | sort -k1,1 -k2,2n > ${name}.bed
    python /Users/guertinlab/pyscripts/BedFix.py ${name}.bed $name.fixed.bed
    readDepth=$(samtools view -c -F 0x40 ${name}.rmdup.bam)
    scale=$(bc <<< "scale=4 ; 10000000.0 / $readDepth")
    echo readDepth
    echo $readDepth
    echo scaleFactor
    echo $scale
#this is a way to be able to upload files directly
#but one can get server space from UVA to store bigWigs, which is recommended.
#I had to reduce the number of lines in the file to <50M, but to get the bigWig just remove the grep and head command
#and use wigToBigWig or bedGraphToBigWig and UCSC has a bigWig scaling tool.
    genomeCoverageBed -scale ${scale} -bg -trackline \
-trackopts 'track type=bedGraph name='"${name}"'.combined.chr1 alwaysZero=on visibility=full' \
-i ${name}.bed -g mm10.chrom.sizes | grep -v chrU | grep -v random | \
grep chr1 > ${name}_sorted_window_scaled_chr1s.bedGraph
    echo word count chr1
    wc -l ${name}_sorted_window_scaled_chr1s.bedGraph
    genomeCoverageBed -scale ${scale} -bg -trackline \
-trackopts 'track type=bedGraph name='"${name}"'.combined.2 alwaysZero=on visibility=full' \
-i ${name}.bed -g mm10.chrom.sizes | grep -v chrU | grep -v random | \
grep -v chr1 > ${name}_sorted_window_scaled_chr2_9s.bedGraph
    echo word count chr2_9
    wc -l ${name}_sorted_window_scaled_chr2_9s.bedGraph
    gzip ${name}_sorted_window_scaled_chr1s.bedGraph
    gzip ${name}_sorted_window_scaled_chr2_9s.bedGraph
done
```

2 Peak calling with `hotspot`

`Hotspot` is not published as a software package, it is described here: (John *et al.*, 2011). This software is incredibly complicated in terms of dependencies and compiling. `Hotspot` is also slow and intermediate

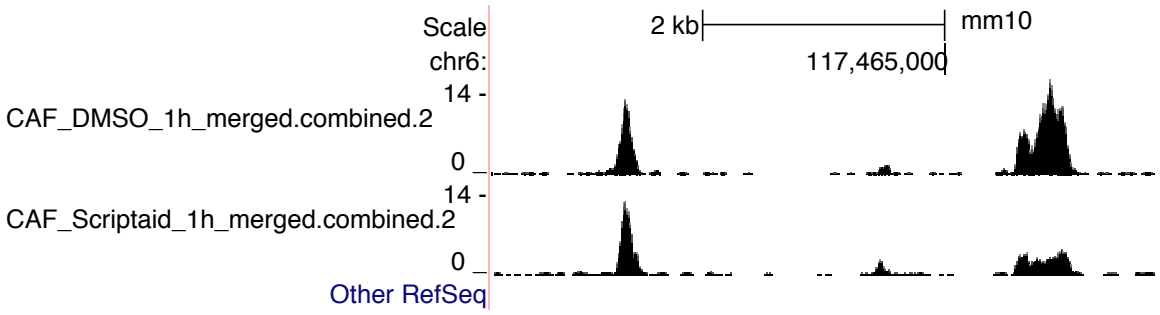


Figure 1: This window includes a peak that is compromised upon 1 hour HDAC inhibition.

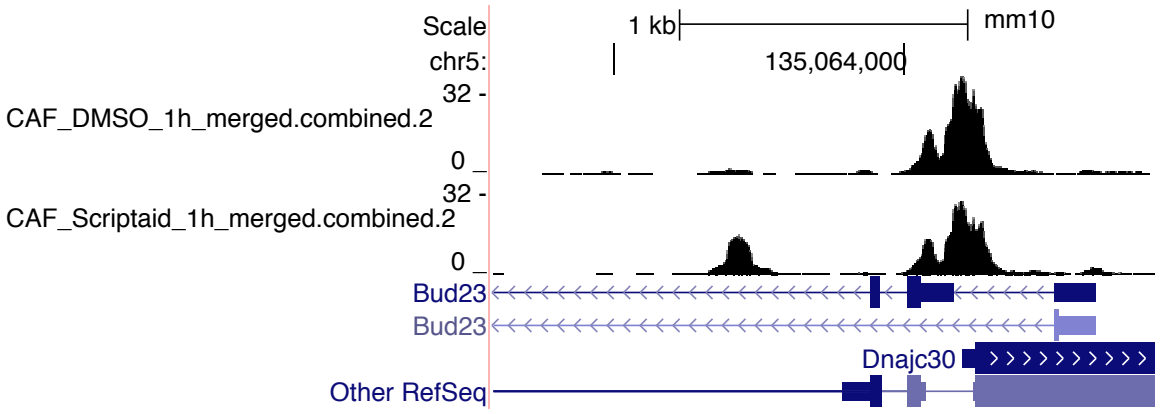


Figure 2: This window shows a peak that is absent in the control.

file sizes are huge. However, 5 years ago I benchmarked `hotspot`, `MACS2`, and `F-seq` and I found that `hotspot` gave the best results. I hope toward the end of the rotation that Tommy and Arun can work together to start implementing these peak callers and thinking about ways to bench mark them using ChIP-seq data and motif callers.

2.1 Generation of `hotspot` input files

I wrote some *ad hoc* `Python` code to automate the peak calling process with `hotspot`. This is not straightforward. The only two files that you would need to change in the future are `180522_atac.runhotspot` and `180522.mm10.arun.runall.tokens.atac.txt`. However, these files are sufficient for all mouse peak calling for this project. For future hg38 projects, I will concatenate the badspots BED file with the blacklisted regions for hg38. notes:
To get `hotspot` to run I changed the wavelets binary available with the source hotspot download to this wavelets source: `http://staff.washington.edu/dbp/WMTSA/NEPH/wavelets.html`.
I also changed line 44 of run_10kb_counts from:

```
      | cut --output-delimiter=" " -f1,2,4 - \
#to

      | cut -d " " -f1,2,4 - \
```

I have this written as a loop below, but you can run it on a pan-merged duplicate removed BAM file.

```
for bam in *_merged.rmdup.bam
do
    name=$(echo $bam | awk -F"/" '{print $NF}' | awk -F".rmdup.bam" '{print $1}')
    echo filename:
    echo $name
    python /Users/guertinlab/pyscripts/runhotspot_all.py $bam 180619_atac.runhotspot
    chmod +x $name.runhotspot
    python /Users/guertinlab/pyscripts/token_hotspot_files.py $bam 180619.mm10.arun.runall.tokens.atac.txt
```

```
    ./$name.runhotspot
done

#Example of combining peaks with p values
for condition in *_merged.rmdup-final
do
    name=$(echo $condition | awk -F"_merged.rmdup-final" '{print $1}')
    cd $condition
    paste ${name}_merged.rmdup.fdr0.01.pks.bed ${name}_merged.rmdup.fdr0.01.pks.pval.txt \
       > ${name}_merged.rmdup.fdr0.01.pks.pval.bed
    cd ..
done

#Example of choosing a p value cutoff based on number of resulting peaks (try several cutoffs)
cd CAF_DMSO_1h_merged.rmdup-final
awk '{ if ($4 < 1e-40) { print } }' CAF_DMSO_1h_merged.rmdup.fdr0.01.pks.pval.bed | wc -l

#Only using peaks with small p values (using the cutoff you chose above)
awk '{ if ($4 < 1e-40) { print } }' CAF_DMSO_1h_merged.rmdup.fdr0.01.pks.pval.bed \
       > CAF_DMSO_1h_merged.rmdup.fdr0.01.pks.pval.smallpval.bed
cd ..

#Merging peaks (need to paste into one line in the terminal)
#You can either use the pks.pval.smallpval.bed files you made or just the pks.bed files
cat CAF_DMSO_1h_merged.rmdup-final/CAF_DMSO_1h_merged.rmdup.fdr0.01.pks.pval.smallpval.bed \
    CAF_DMSO_7d_merged.rmdup-final/CAF_DMSO_7d_merged.rmdup.fdr0.01.pks.pval.smallpval.bed \
    CAF_Scriptaid_1h_merged.rmdup-final/CAF_Scriptaid_1hr.rmdup.fdr0.01.pks.pval.smallpval.bed \
    CAF_Scriptaid_7d_merged.rmdup-final/CAF_Scriptaid_7d_merged.rmdup.fdr0.01.pks.pval.smallpval.bed \
    | sort -k1,1 -k2,2n | awk ' $2 >= 0 ' | mergeBed -i stdin | \
    intersectBed -wa -v -a stdin -b mm10.blacklist.bed > CAF_hotspot_merged_peaks_smallpval.bed
```

3 Peak calling with macs2

We will use macs2 to call peaks (Zhang *et al.*, 2008).

3.1 Running macs2

We will exclude blacklisted regions, which you can read about. ENCODE classified these regions as problematic for ChIP-seq and other assays. I suspect that you will find that the set of differential accessible regions won't change a lot with different sets of peaks. The reason for this is because DESeq2 uses a shrinkage factor to correct for large fold change differences that result from low read counts. For example, going from 1 to 2 reads in a region is not equivalent to going from 100 to 200 reads in a region despite a 2-fold change in both. Therefore, I suspect the low intensity peaks that are variable between peak calling parameters will rarely have enough reads to have a meaningful fold change and their shinkage prior correction factors will ensure these don't make it into the significantly changes peaks.

```
for bam in CAF*merged.rmdup.bam
do
    name=$(echo $bam | awk -F"_merged.rmdup.bam" '{print $1}')
    echo $name
    #keep duplicates becasue I already filtered them, over 50 is rare and likely mapping errors
    #you probably want to adjust the -q and -m argument to get an average of between 100k and 200k peaks per replicate
    macs2 callpeak -t $bam -f BAMPE -n ${name} --outdir ${name}_macs \
  -g mm -B --call-summits --keep-dup 50 -q 0.05 -m 10 200
    cd ${name}_macs
    awk '{OFS="\t";} {print $1,$2-100,$3+100,$4,$5}' ${name}_summits.bed > ../${name}_summit_window.bed
    cd ..
done

#Example of only using peaks with small p values to keep each set under 200,000
awk '{ if ($5 > 6) { print } }' CAF_DMSO_1h_summit_window.bed > CAF_DMSO_1h_summit_window_pval_smallpval.bed

#Merging and removing blacklisted sites
wget http://mitra.stanford.edu/kundaje/akundaje/release/blacklists/mm10-mouse/mm10.blacklist.bed.gz
gunzip mm10.blacklist.bed.gz
#You can do this with the original files or the small pval files
cat *_summit_window.bed | sort -k1,1 -k2,2n | \
    awk ' $2 >= 0 ' | \
    mergeBed -i stdin | \
    intersectBed -wa -v -a stdin -b mm10.blacklist.bed > CAF_merged_peaks.bed
```

4 Analysis in R

Working in R with the data frames, performing exploratory analysis, and plotting the data is the highlight of analysis. Powerful syntax to learn is how to select rows based on certain column values– we often label these row with a categorical variable and add this as a column. I am happy to go over this with you, but the second and third lines of the function `process.deseq.df` are a good example, which selects rows in `ma.df` that have an FDR adjusted p-value, `padj`, below the argument specification, this value also cannot be NA, and the fold change column is greater than or less than the `fc` argument provided.
After figuring out what comparisons to make, we perform statistical analyses and graphically represent the quantitative data.
Unless you can get `hotspot` to work or you start implementing other peak callers, you should use *CAF_Scriptaid_1hr.rmdup.fdr0.01.pks.pval.smallpval.bed* as a set of preliminary peaks. Note that these were called by merging 1 hr. SA samples, so we are not equipped to detect changes that ablate accessibility. In Figure 1 the compromised peak was still detected after SA treatment, but you may be able to cherry pick a different peak if you use the DMSO treatment to call differential accessibility. For these reasons in finalized peak files we will merge peaks from many conditions and operate exclusively on this merged file.
Now you have the ability to call peaks using `macs2` with several different arguement parameters and run through these analyses and see if you get the same results over a range of different arbitrary `-q` and `-m` values.

4.1 Principle Components Analysis of replicates and conditions

I performed principle components analysis and replicates and conditions separated as we expected. This is useful for identifying anomalies and mislabeled files.

```
library(DESeq2)
library(lattice)
library(bigWig)

#functions
get.raw.counts.interval <- function(df, path.to.bigWig, file.prefix = 'M') {
    df = df[,1:3]
    vec.names = c()
    inten.df=data.frame(matrix(ncol = 0, nrow = nrow(df)))

    for (mod.bigWig in Sys.glob(file.path(path.to.bigWig,
                                          paste(file.prefix, "*.bigWig", sep =''))) {
        factor.name = strsplit(strsplit(mod.bigWig, "/")[[1]][length(strsplit(mod.bigWig, "/")[[1]])], '\\.')[[1]][1]
        print(factor.name)
        vec.names = c(vec.names, factor.name)
        loaded.bw = load.bigWig(mod.bigWig)
        mod.inten = bed.region.bpQuery.bigWig(loaded.bw, df)
        inten.df = cbind(inten.df, mod.inten)
    }
    colnames(inten.df) = vec.names
    r.names = paste(df[,1], ':', df[,2], '-', df[,3], sep='')
    row.names(inten.df) = r.names
    return(inten.df)
}

#two - six  replicates
plotPCAlattice <- function(df, file = 'PCA_lattice.pdf', reps = 3) {
    perVar = round(100 * attr(df, "percentVar"))
#identical replicate color, transparency difference
    if (reps == 2) {
    colpal = sort(c(rainbow(nrow(df)/2, start = 0.01, end = 0.8, alpha = 0.9),
                    rainbow(nrow(df)/2, start = 0.01, end = 0.8, alpha = 0.7)))
    }
    if (reps == 3) {
        colpal = sort(c(rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.9),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.7),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.5)))
    }
    if (reps == 4) {
        colpal = sort(c(rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.9),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.75),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.60),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.45)))
    }
    if (reps == 5) {
        colpal = sort(c(rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.9),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.75),
```

```
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.60),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.45),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.30)))
    }
    if (reps == 6) {
        colpal = sort(c(rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.9),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.75),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.60),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.45),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.30),
                    rainbow(nrow(df)/3, start = 0.01, end = 0.8, alpha = 0.15)))
        }
    df$group = gsub('_', ' ', df$group)
    pdf(file, width=6, height=6, useDingbats=FALSE)
    print(xyplot(PC2 ~ PC1, groups = group, data=df,
           xlab = paste('PC1: ', perVar[1], '% variance', sep = ''),
           ylab = paste('PC2: ', perVar[2], '% variance', sep = ''),
           par.settings = list(superpose.symbol = list(pch = c(20), col=colpal)),
           pch = 20,
           auto.key = TRUE,
           col = colpal))
    dev.off()
}

run.deseq.table <- function(mat) {
  sample.conditions = factor(colnames(mat))
  deseq.counts.table = DESeqDataSetFromMatrix(mat, DataFrame(sample.conditions), ~ sample.conditions);
  colData(deseq.counts.table)$condition<-factor(colData(deseq.counts.table)$sample.conditions, levels=colnames(mat));
  dds = DESeq(deseq.counts.table);
  return(dds)
}

#processes
directory = "/Users/guertinlab/Desktop/Dudley_Lab/"
setwd(directory)

#Change the file to the one you want to read
CAF.file = read.table("CAF_merged_peaks.bed")

df.CAF = get.raw.counts.interval(CAF.file, directory, file.prefix = 'CAF')

#PCA--see DESeq2 vignette for explanation
#I developed the plotting function because I prefer lattice and grid graphics
df.caf.deseq.raw = run.deseq.table(df.CAF)
rld_caf = rlogTransformation(df.caf.deseq.raw)
pca.dat = plotPCA(rld_caf, intgroup="condition", returnData=TRUE)
percentVar = round(100 * attr(pca.dat, "percentVar"))
#Change the file name
plotPCAlattice(pca.dat, file = 'PCA_macs2_CAF_lattice_guertin.pdf', reps = 3)
```

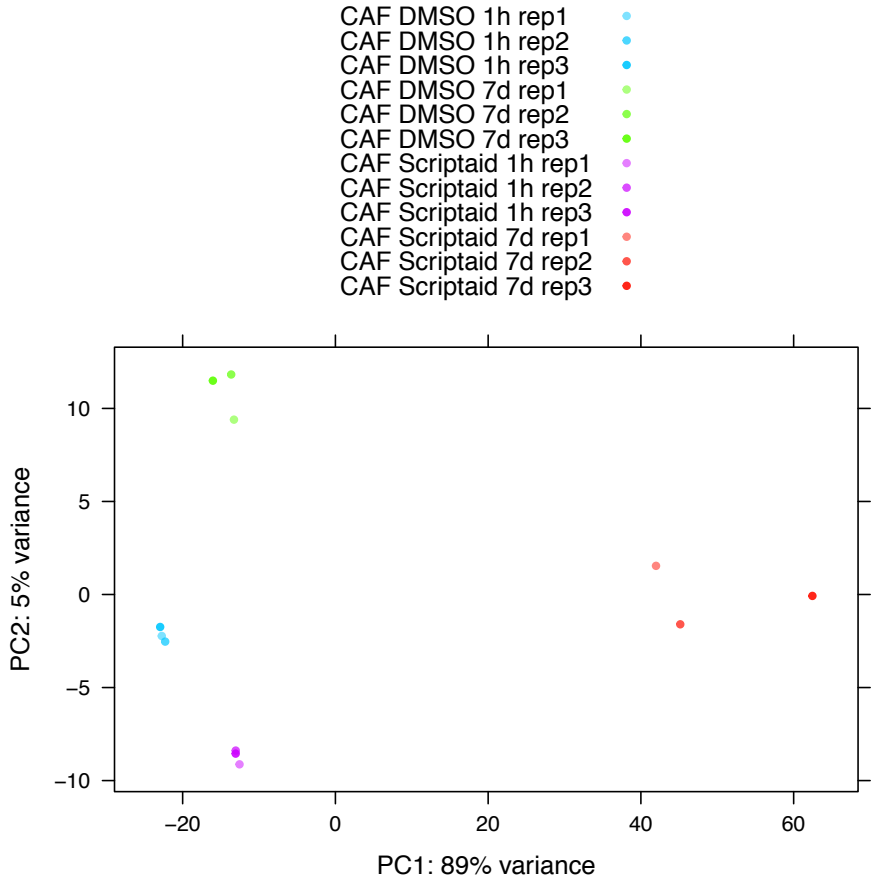


Figure 3: Principle components analysis clusters replicates and conditions as we expected.

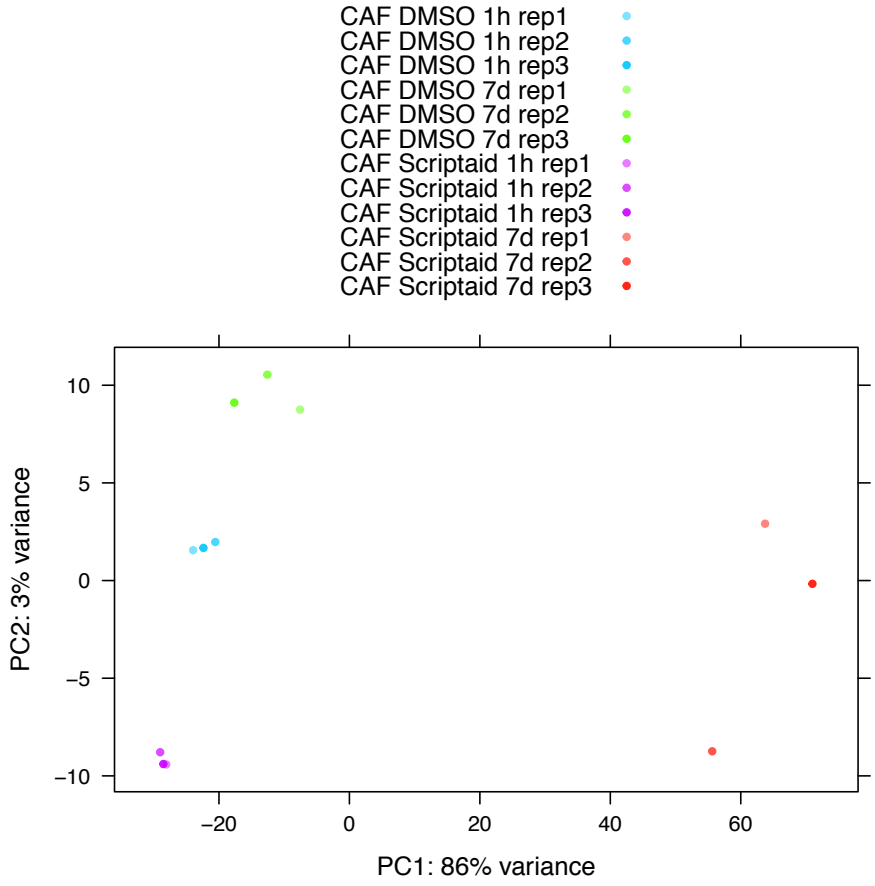


Figure 4: PCA using peaks called with MACS2 on all the samples.

4.2 Differential Accessibility + 1hr. DMSO vs. ScriptAid

The example code performs differenetial expression with DESeq2. In addition to peak calling and motif analysis, differential expression algorithms are a step that can be optimized. I recently compared `edgeR`, `DESeq2`, and `Cuffdiff` and I prefer DESeq2. Chongzhi Zang's lab independently compared a bunch of differential expression software and he came to the same conclusion.
Last time I ran the hotspot program I ran into memory issues (I think because Illustrator was merging points). I have not run the hotspots again, because this file is fine for preliminary analysis in the grant we are submitting: CAF_Scriptaid_1hr.rmdup.fdr0.01.pks.pval.smallpval.bed

```
#I am not loading libraries because I assume you have the same R session as the previous code chunk

#functions
run.deseq.list <- function(mat, untreatedreps = 3, treatedreps = 3) {
    sample.conditions = factor(c(rep("untreated", untreatedreps) , rep("treated", treatedreps)))
    deseq.counts.table = DESeqDataSetFromMatrix(mat, DataFrame(sample.conditions), ~ sample.conditions);
    colData(deseq.counts.table)$condition<-factor(colData(deseq.counts.table)$sample.conditions,
                                                  levels=c("untreated","treated"));
    dds = DESeq(deseq.counts.table);
    res = results(dds);
    res = res[order(res$padj),];
    return(res)
}

plot.ma.lattice <- function(ma.df, filename = 'file.name', p = 0.01,
                            title.main = "Differential Transposase",log2fold =0.0)
  {
  pdf(paste("MA_plot", filename, ".pdf", sep=''), width=3.83, height=3.83, useDingbats=FALSE);
  print(xyplot(ma.df$log2FoldChange ~ log(ma.df$baseMean, base=10),
               groups=(ma.df$padj < p & abs(ma.df$log2FoldChange) > log2fold & !is.na(ma.df$padj)),
               col=c("black","red"), main=title.main, scales="free", aspect=1, pch=20, cex=0.5,
               ylab=expression("log"[2]~"Transposition fold change"),
               xlab=expression("log"[10]~"Mean of Normalized Counts"),
               par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                                 par.ylab.text=list(cex=1.1,font=2))))
  dev.off()
}

plot.ma.lattice.publishable <- function(ma.df, filename = 'file.name', adj = 0.001,
                                        title.main = "Differential Expression") {
    ma.df$group <- 'a'
    ma.df[!is.na(ma.df$padj) & !is.na(ma.df$log2FoldChange) &
          ma.df$padj < adj & ma.df$log2FoldChange < 0,]$group <- 'b'
    ma.df[!is.na(ma.df$padj) & !is.na(ma.df$log2FoldChange) &
          ma.df$padj < adj & ma.df$log2FoldChange > 0,]$group <- 'c'
    ma.df[ma.df$padj > 0.5 & !is.na(ma.df$padj) &
                        !is.na(ma.df$log2FoldChange) &
                        abs(ma.df$log2FoldChange) < 0.25,]$group <- 'd'
    pdf(paste("MA_plot_", filename, "_FDR_pval_", as.character(adj),".pdf", sep=''),
        width=3.83, height=3.83, useDingbats=FALSE)
    print(xyplot(ma.df$log2FoldChange ~ log(ma.df$baseMean, base=10), groups=ma.df$group,
                 col=c("grey85","#2290cf","#ce228e", "grey65"), main=title.main, scales="free", aspect=1,
                 pch=20, cex=0.3, ylab=expression("log"[2]~"Differential ATAC"),
                 xlab=expression("log"[10]~"Mean of Normalized Counts"),
                 par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                                   par.ylab.text=list(cex=1.1,font=2))))
    dev.off()
    return(ma.df)
}

#files for MEME. it needs to be polished
#this should to have a nested function
#In the FIMO section I cleaned this up...see what I did for an example
process.deseq.df <- function(ma.df, filename = 'file.name', adj = 0.001, pval = NA, fc = 0) {
    decreased =  ma.df[ma.df$padj < adj & !is.na(ma.df$padj) & ma.df$log2FoldChange < fc,];
    increased =  ma.df[ma.df$padj < adj & !is.na(ma.df$padj) & ma.df$log2FoldChange > fc,];
    not.different =  ma.df[ma.df$padj > 0.5 & !is.na(ma.df$padj) & !is.na(ma.df$log2FoldChange) &
                           abs(ma.df$log2FoldChange) < 0.25,];
    coor = rownames(decreased)
    coor.start = sapply(strsplit(sapply(strsplit(as.character(coor),':'), "[", 2), "-"), "[", 1);
    coor.end = sapply(strsplit(as.character(coor),'-'), "[", 2)
    coor.chr = sapply(strsplit(as.character(coor),':'), "[", 1)
    df.coor = cbind(coor.chr, coor.start, coor.end, as.character(decreased$baseMean),
                    as.character(decreased$log2FoldChange), '+', as.character(decreased$lfcSE),
                    as.character(decreased$pvalue), as.character(decreased$padj))
    write.table(df.coor, file = paste('decreased_', filename, '_', adj, '_FDR.bed', sep =''),
                sep = '\t', quote=F, row.names=F, col.names=F)

    coor = rownames(increased)
    coor.start = sapply(strsplit(sapply(strsplit(as.character(coor),':'), "[", 2), "-"), "[", 1);
    coor.end = sapply(strsplit(as.character(coor),'-'), "[", 2)
    coor.chr = sapply(strsplit(as.character(coor),':'), "[", 1)
    df.coor = cbind(coor.chr, coor.start, coor.end, as.character(increased$baseMean),
```

```
                    as.character(increased$log2FoldChange), '+', as.character(increased$lfcSE),
                    as.character(increased$pvalue), as.character(increased$padj))
    write.table(df.coor, file = paste('increased_', filename, '_', adj, '_FDR.bed', sep =''),
                sep = '\t', quote=F, row.names=F, col.names=F)

    coor = rownames(not.different)
    coor.start = sapply(strsplit(sapply(strsplit(as.character(coor),':'), "[", 2), "-"), "[", 1);
    coor.end = sapply(strsplit(as.character(coor),'-'), "[", 2)
    coor.chr = sapply(strsplit(as.character(coor),':'), "[", 1)
    df.coor = cbind(coor.chr, coor.start, coor.end, as.character(not.different$baseMean),
                    as.character(not.different$log2FoldChange), '+', as.character(not.different$lfcSE),
                    as.character(not.different$pvalue), as.character(not.different$padj))
    write.table(df.coor, file = paste('notDifferent_', filename, '_', adj, '_FDR.bed', sep =''),
                sep = '\t', quote=F, row.names=F, col.names=F)
}

#processes
df.CAF.sa.1h.deseq = run.deseq.list(df.CAF[,c(
    'CAF_DMSO_1h_rep1',
    'CAF_DMSO_1h_rep2',
    'CAF_DMSO_1h_rep3',
    'CAF_Scriptaid_1h_rep1',
    'CAF_Scriptaid_1h_rep2',
    'CAF_Scriptaid_1h_rep3')], untreatedreps = 3, treatedreps = 3)
df.CAF.sa.7d.deseq = run.deseq.list(df.CAF[,c(
    'CAF_DMSO_7d_rep1',
    'CAF_DMSO_7d_rep2',
    'CAF_DMSO_7d_rep3',
    'CAF_Scriptaid_7d_rep1',
    'CAF_Scriptaid_7d_rep2',
    'CAF_Scriptaid_7d_rep3')], untreatedreps = 3, treatedreps = 3)

#Change names
plot.ma.lattice(df.CAF.sa.1h.deseq, filename = 'CAF_macs2_1h_ScriptAid_0.0001', p=0.0001, title.main = 'Differential ATAC')
plot.ma.lattice(df.CAF.sa.7d.deseq, filename = 'CAF_macs2_7d_ScriptAid_0.0001', p=0.0001, title.main = 'Differential ATAC')
df.CAF.sa.1h.deseq.grps = plot.ma.lattice.publishable(df.CAF.sa.1h.deseq, 'df_macs2_CAF_sa_1h_deseq',
                                                      adj = 0.0001, title.main = '')
df.CAF.sa.7d.deseq.grps = plot.ma.lattice.publishable(df.CAF.sa.7d.deseq, 'df_macs2_CAF_sa_7d_deseq',
                                                      adj = 0.0001, title.main = '')

#Make coordinate files for MEME
process.deseq.df(df.CAF.sa.1h.deseq, filename = 'df.macs2.CAF.sa.1h.deseq', adj = 0.0001, fc = 0)
process.deseq.df(df.CAF.sa.7d.deseq, filename = 'df.macs2.CAF.sa.7d.deseq', adj = 0.0001, fc = 0)

#Save more than one:
save(df.CAF, df.CAF.sa.1h.deseq, df.CAF.sa.1h.deseq.grps,
     df.CAF.sa.7d.deseq, df.CAF.sa.7d.deseq.grps,
     file = 'df.macs2.CAF.all.Rdata')
```

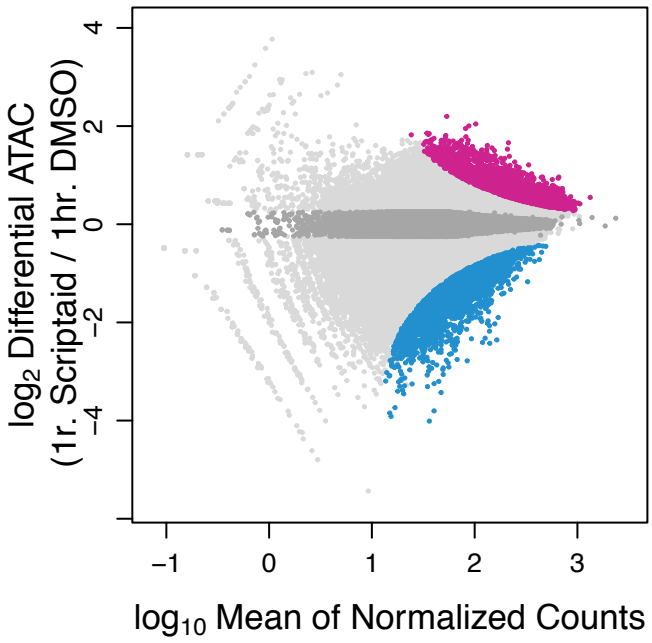


Figure 5: We used an FDR of 0.0001 to identify differentially accessible ATAC peaks. 3329 peaks decrease their intensity; 5774 increase their intensity.

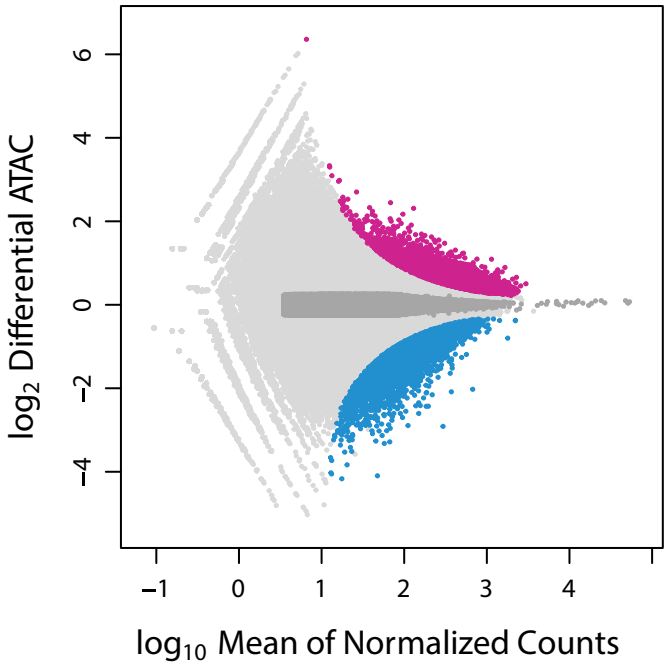


Figure 6: MACS2-called peaks on pooled samples at 1 hour. 6009 peaks decrease their intensity; 6000 increase their intensity.

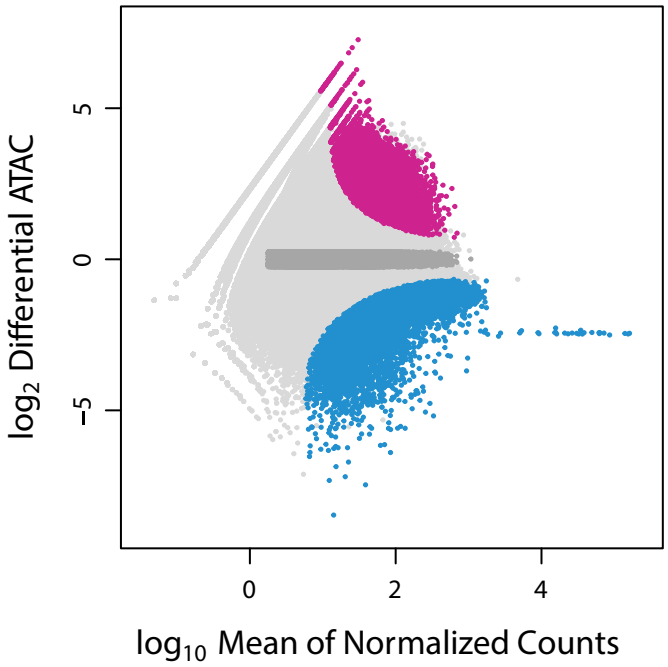


Figure 7: MACS2-called peaks on pooled samples at 7 days. 10588 peaks decrease their intensity; 16926 increase their intensity.

5 Motif Analysis

Along with peak calling, various software used for motif analysis should be sysmtatically compared for run time and sensitivity. Sensistivity is easy becasue the presence of additional known motifs is a straightforward indication tha tthe algorithm is finding a motif. Below we are using MEME (Bailey *et al.*, 2006), but it doesn't perform differential motif analysis, which would be powerful and we could compare the different classes of accessible regions.

5.1 MEME

I am using bedtools to get the FASTA sequence and MEME-chip to perform *de novo* motif analysis.

```
for bed in *deseq_1e-04_FDR.bed
do
	echo $bed
	name=$(echo $bed | awk -F'.bed' '{print $1}')
	fastaFromBed -fi mm10.fa -bed $name.bed -fo $name.fasta
done

for fasta in decreased*1e-04_FDR.fasta
do
	echo $fasta
	name=$(echo $fasta | awk -F'.fasta' '{print $1}' | awk -F'decreased_' '{print $2}')

	meme-chip -o decreased_$name.meme_chip_output -dna -meme-nmotifs 15 -meme-minw 5 -meme-maxw 15 \
		-filter-thresh 0.01 -fimo-skip -spamo-skip -neg increased_$name.fasta decreased_$name.fasta
	meme-chip -o increased_$name.meme_chip_output -dna -meme-nmotifs 15 -meme-minw 5 -meme-maxw 15 \
		-filter-thresh 0.01 -fimo-skip -spamo-skip -neg decreased_$name.fasta increased_$name.fasta
done
```

I manually looked at these motif and used the resultant HTML file and TOMTOM to identify transcription factor target motif families that are unique to peaks that increase or decrease, but not the other class. In the decreased class I saw motif 5 is ZNF143-like, but it may be novel. Motifs 7 and 8 are RFX and NR4A families, respectively. Motif 9 is TCF-like, but it may be novel. This potential novelty is premature, I just did a cursory query to some data bases of known TF motifs. For the peaks that increase I see motif 1 is TEAD family and motif 2 is a true ZNF143 motif. Are there two ZNF143 genes with slightly diverged DNA binding domains in mouse?

Results from merged MACS2 peaks (using MEME 4.12): In the decreased class I saw motif 1 is similar to the SP1/KLF family. Motif 4 is CTCF-like. Motif 10 is NR5A2-like. Motif 15 is the NR4A-like motif that you found. For the peaks that increase I see motif 1 is TEAD family and motif 4 is a ZNF143 motif. Two motifs are similar to the SP2/KLF family motif you found, but I wasn't convinced. Both classes had some JUN/FOS motifs.

At 7 days: In the decreased class motifs 1 and 7 are similar to the SP1/KLF family. Motif 2 is CTCF-like. Motif 4 is NFY-like. For the peaks that increase, motif 1 is in the JUN/FOS family. Motif 2 is RUNX1-like. Motif 3 is Tead family. Motif 5 is EBF1-like.

MEME 5.0 results: Decreased at 1 hour: Motif 10 is TFAP4-like. Increased at 1 hour: Motif 1 is TEAD family. Motif 12 is slightly E2F6-like. Decreased at 7 days: Motif 2 is SP/KLF family. Motif 6 is CTCF-like. Motif 7 is ZNF143-like. Motif 10 is ZFP691-like. Motif 12 is MEIS family. Motif 14 is maybe ETV family. Increased at 7 days: Motif 1 is JUN/FOS family.

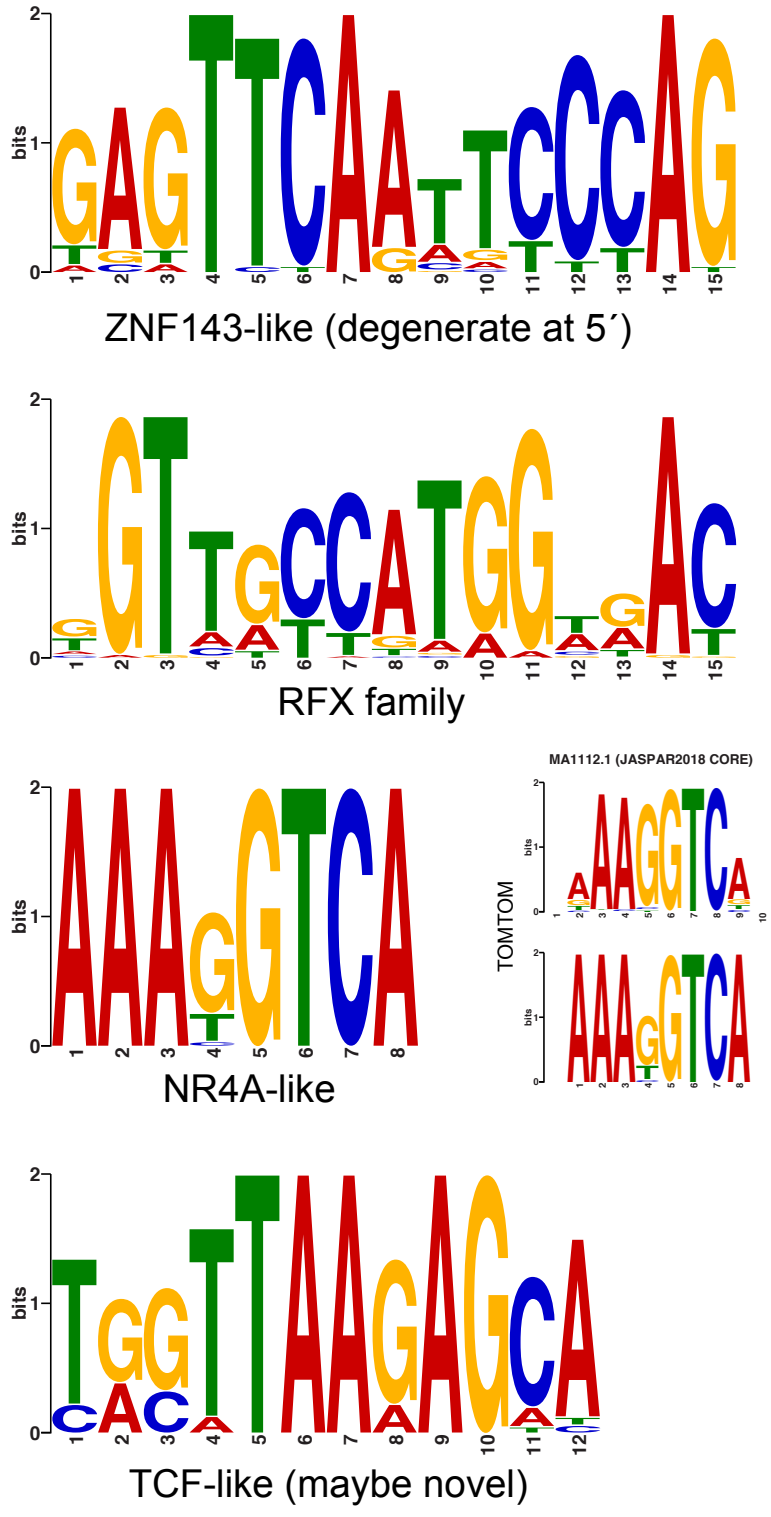


Figure 8: These motifs represent motif families that are unique to peaks that decrease intensity upon treatment. The ZNF143-like and TCF-like don't conform well to the motif data base, but this could have technical or biological meaning.

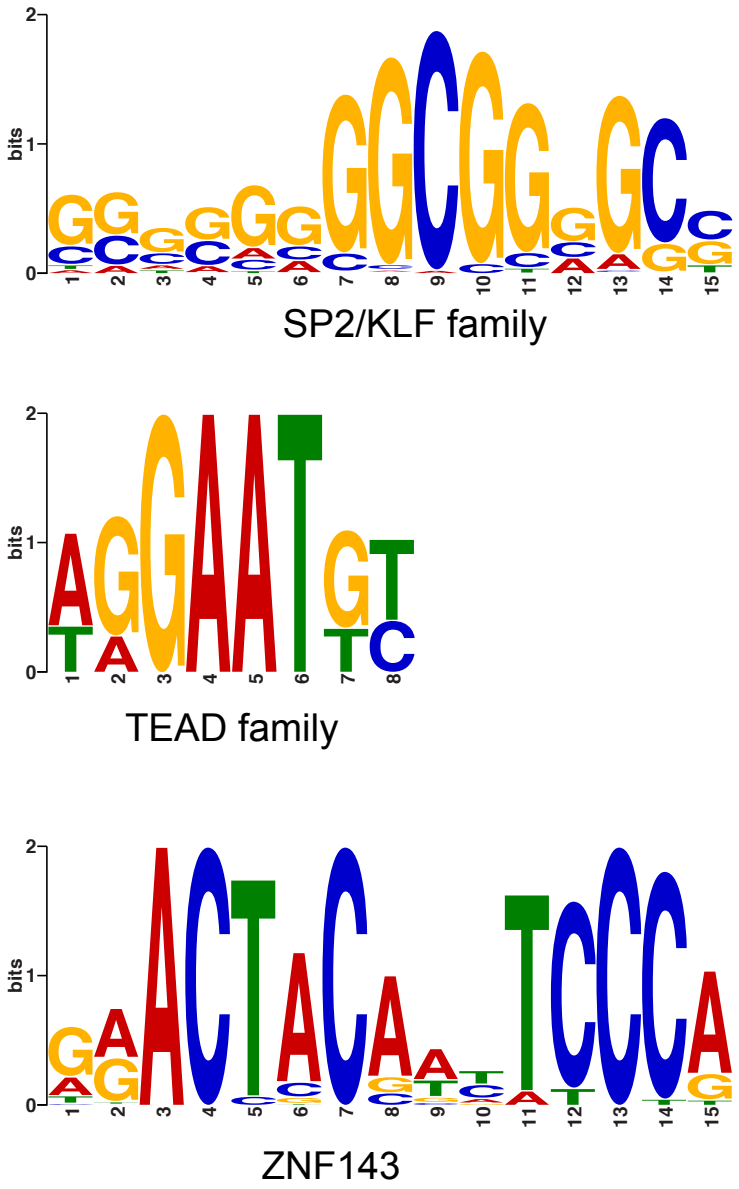
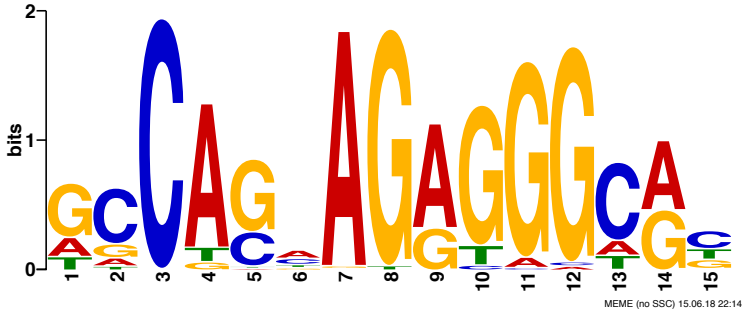


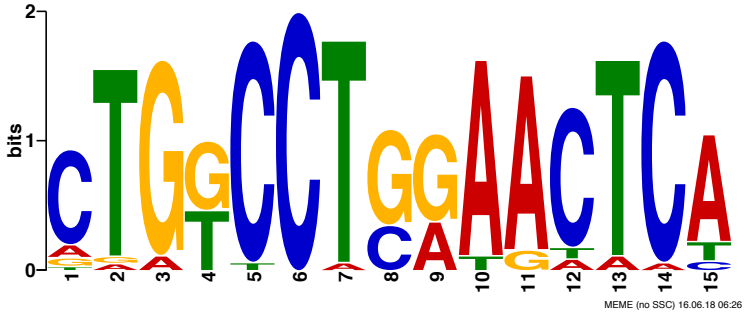
Figure 9: These motifs represent motif families that are found *de novo* exclusively in peaks that increase intensity upon treatment.



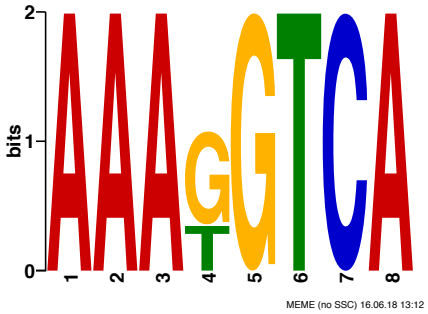
SP1/KLF family



CTCF-like

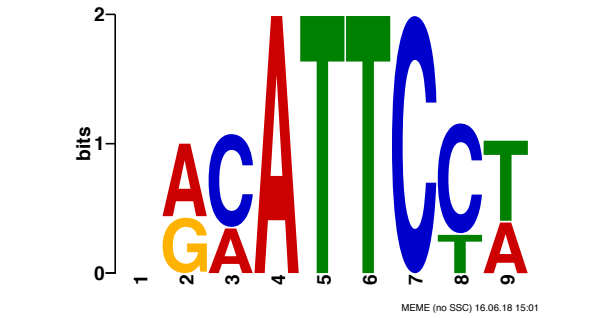


NR5A2-like

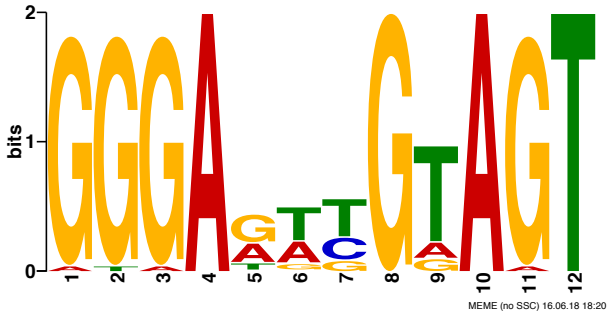


NR4A-like

Figure 10: These motifs represent motif families that are unique to MACS2-called peaks that decrease intensity upon 1 hour treatment.



TEAD family



ZNF143-like

Figure 11: These motifs represent motif families that are unique to MACS2-called peaks that increase intensity upon 1 hour treatment.

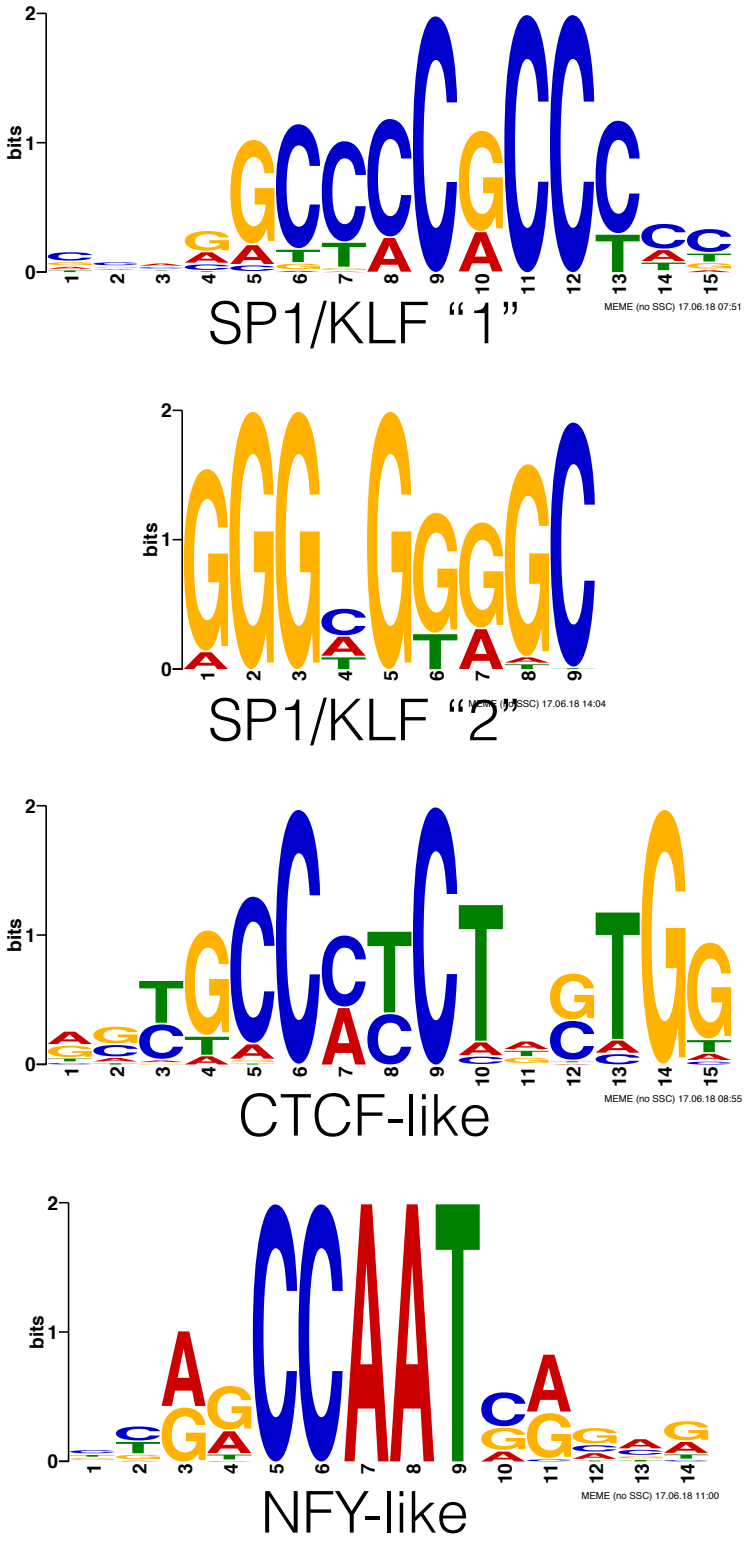


Figure 12: These motifs represent motif families that are unique to MACS2-called peaks that decrease intensity upon 7 day treatment.

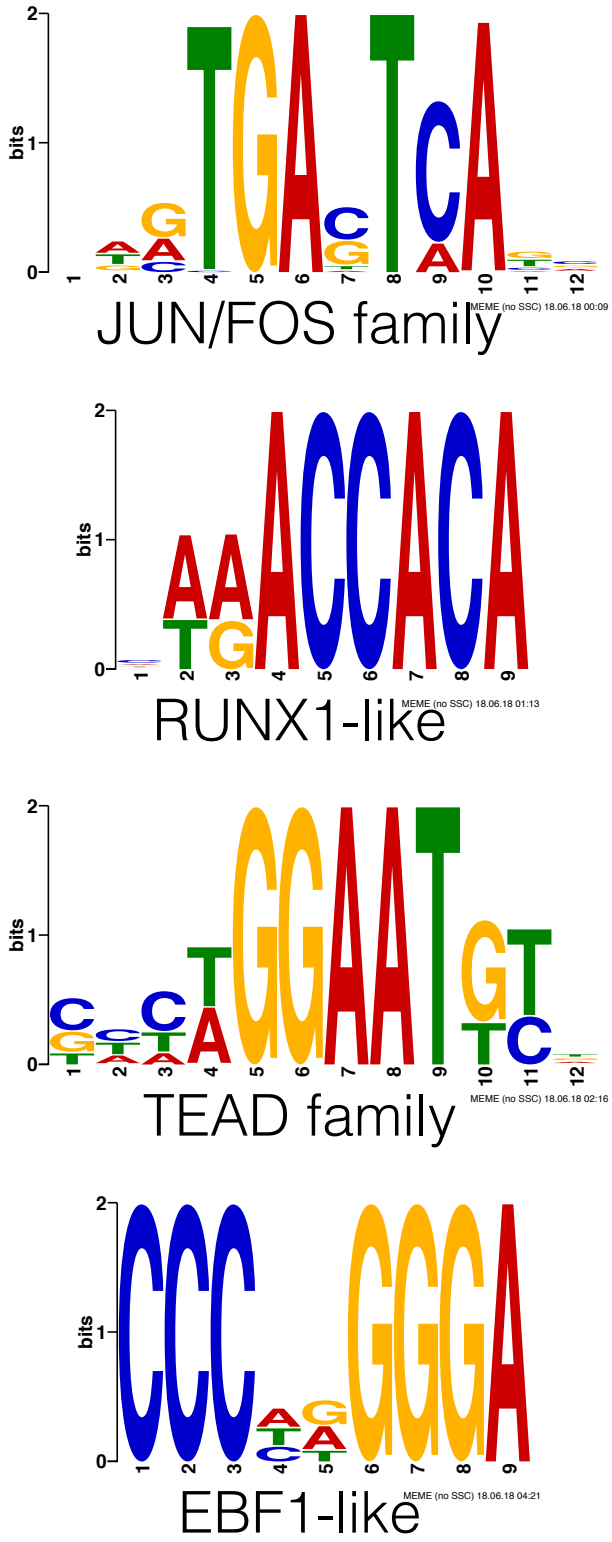


Figure 13: These motifs represent motif families that are unique to MACS2-called peaks that increase intensity upon 7 day treatment.

5.2 FIMO

Next we will map motif instances around hypersensitivity summits. This is more sensitive and we are looking for specific enrichment (signified by a peak) in either the increased or decreased upon treatment class, but not both. The unchanged control peaks may be intermediate.

To ensure robust results one should use the *de novo* identified motif and a matched motif from data bases. Also vary the threshold, see an example in the code chunk.
I only explored a few motifs, but exhaustively characterizing the motifs identified with DREME, MEME, and other *de novo* motif callers is necessary for Arun's project. We can adopt previously published methods to group and query motifs, then integrate expression data to get candidate factors within families that may bind the motifs (Liu *et al.*, 2017).

```
fimo --max-stored-scores 1000000 -motif KGTTGCCATGGWRAC -o RFX_mm10_de_novo.fimo \
decreased.FDR_0.0001.in.CAF.1hr.meme_chip_output/meme_out/meme.txt mm10.fa
#I changed the threshold on this because invariant weights are too strict
fimo --thresh 1e-3 --max-stored-scores 3000000 -motif AAAGGTCA -o NR4A_mm10_de_novo_invariant_positions.fimo \
decreased.FDR_0.0001.in.CAF.1hr.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif WGGAATGY -o TEAD_mm10_de_novo.fimo \
increased.FDR_0.0001.in.CAF.1hr.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif TGGGARWTGTAGTYC -o ZNF143_mm10_de_novo.fimo \
increased.FDR_0.0001.in.CAF.1hr.meme_chip_output/meme_out/meme.txt mm10.fa
#I changed the threshold here, because of the high frequency of this motif put it off the y-axis when plotting.
#Before concluding anything, you need to test a range of threshold anyways
fimo --thresh 1e-6 --max-stored-scores 1000000 -motif GSGSGGGGCGGGGCS -o SP2_mm10_de_novo.fimo \
increased.FDR_0.0001.in.CAF.1hr.meme_chip_output/meme_out/meme.txt mm10.fa

#I still don't like the invariant positions in the NR4A matrix, so I will use the JASPAR motif
#plus it was taking u too much memory to relax the threshold
wget http://jaspar2018.genereg.net/api/v1/matrix/MA1112.1.meme
fimo --max-stored-scores 1000000 -motif MA1112.1 -o NR4A_mm10_de_novo.fimo \
MA1112.1.meme mm10.fa

#Tommy's motifs
#1h increased
fimo --max-stored-scores 1000000 -motif WGGAATGY -o TEAD_1h_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif GGGARWYGTAGT -o ZNF143_1h_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
#1h decreased
fimo --max-stored-scores 1000000 -motif GKGGGHGGGGCB -o SP1_1h_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif GCCAGMAGAGGGCRB -o CTCF_1h_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif CTGKCCTGRAACTCA -o NR5A2_1h_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif AAAGGTCA -o NR4A_1h_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
#7d increased
fimo --max-stored-scores 1000000 -motif NDRTGASTCAKV -o JUN_7d_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif HWRACCACA -o RUNX1_7d_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif SHYWGGAATGYD -o TEAD_7d_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif CCCHRGGGA -o EBF1_7d_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
#7d decreased
fimo --max-stored-scores 1000000 -motif CSHRGCCCCGCCCCC -o SP1_1_7d_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif GGGMGGGGC -o SP1_2_7d_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif RSYGCCMYCTDSTGG -o CTCF_7d_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif BYRRCCAATSRGMR -o NFY_7d_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa

#MEME 5.0 motifs
#1h decreased
fimo --max-stored-scores 1000000 -motif CAKCTGW -o TFAP4_1h_MEME_5.0_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
#1h increased
fimo --thresh 1e-3 --max-stored-scores 10000000 -motif GGAATG -o TEAD_1h_MEME_5.0_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif CCTCCCTCC -o E2F6_1h_MEME_5.0_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
#7d decreased
fimo --max-stored-scores 1000000 -motif GCCCCRCC -o SP-KLF_7d_MEME_5.0_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif TGCCMYCTDSTGG -o CTCF_7d_MEME_5.0_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif TGGGARWTGTAGTY -o ZNF143_7d_MEME_5.0_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
```

```
fimo --thresh 1e-3 --max-stored-scores 10000000 -motif GMGCAC -o ZFP691_7d_MEME_5.0_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --thresh 1e-2 --max-stored-scores 100000000 -motif TGTCA -o MEIS_7d_MEME_5.0_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
fimo --max-stored-scores 1000000 -motif CGGAAGTGASG -o ETV_7d_MEME_5.0_mm10_de_novo.fimo \
decreased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa
#7d increased
fimo --max-stored-scores 1000000 -motif TGASTCAT -o JUN-FOS_7d_MEME_5.0_mm10_de_novo.fimo \
increased_df.macs2.CAF.sa.7d.deseq_1e-04_FDR.meme_chip_output/meme_out/meme.txt mm10.fa

#Converting to bigWig for analysis in R
for fimo in *_mm10_de_novo.fimo/fimo.tsv
do
    name=$(echo $fimo | awk -F"_mm10_de_novo.fimo" '{print $1}')
    cat $fimo | grep -v sequence_name | cut -f3-5 | sort -k1,1 -k2,2n | \
awk '!seen[$0]++' > ${name}_mm10_instances_1M.bed
    mergeBed -i ${name}_mm10_instances_1M.bed |
awk '{OFS="\t";} {print $1,$2,$3,1}' > ${name}_mm10_instances_1M_one.bed
    bedGraphToBigWig ${name}_mm10_instances_1M_one.bed mm10.chrom.sizes ${name}_mm10_instances.bigWig
done

#Example of trying a few thresholds for SP1
cd SP1_1h_mm10_de_novo.fimo
awk '(NR == 1) || ($8 < 1e-4)' fimo.tsv > fimo_1e-4.tsv
awk '(NR == 1) || ($8 < 1e-5)' fimo.tsv > fimo_1e-5.tsv
awk '(NR == 1) || ($8 < 1e-6)' fimo.tsv > fimo_1e-6.tsv
awk '(NR == 1) || ($8 < 1e-7)' fimo.tsv > fimo_1e-7.tsv
rm fimo.tsv
cd ..

#Example of generating bigWig files for the various SP1 thresholds
for fimo in SP1_1h_mm10_de_novo.fimo/fimo_*.txt
do
    name=$(echo $fimo | awk -F"_mm10_de_novo.fimo" '{print $1}')
    significance=$(echo $fimo | awk -F".txt" '{print $1}' | awk -F"fimo_" '{print $2}')
    cat $fimo | grep -v sequence_name | cut -f3-5 | sort -k1,1 -k2,2n | \
awk '!seen[$0]++' > ${significance}_${name}_mm10_instances_1M.bed
    mergeBed -i ${significance}_${name}_mm10_instances_1M.bed |
awk '{OFS="\t";} {print $1,$2,$3,1}' > ${significance}_${name}_mm10_instances_1M_one.bed
    bedGraphToBigWig ${significance}_${name}_mm10_instances_1M_one.bed mm10.chrom.sizes
        ${significance}_${name}_mm10_instances.bigWig
done
```

```
deseq.df.to.3col.bed <- function(ma.df) {
    coor = rownames(ma.df)
    coor.start = sapply(strsplit(sapply(strsplit(as.character(coor),':'), "[", 2), "-"), "[", 1);
    coor.end = sapply(strsplit(as.character(coor),'-'), "[", 2)
    coor.chr = sapply(strsplit(as.character(coor),':'), "[", 1)
    df.coor = cbind(coor.chr, coor.start, coor.end)
    colnames(df.coor) = c('chr', 'start', 'end')
    df.coor = data.frame(df.coor,  stringsAsFactors = FALSE)
    df.coor[,2] = as.numeric(df.coor[,2])
    df.coor[,3] = as.numeric(df.coor[,3])
    return(df.coor)
}

bed.window <- function(bed, half.window) {
    bed[,2] = (bed[,2] + bed[,3])/2 - half.window
    bed[,3] = bed[,2] + 2 * half.window
    return(bed)
}

get.motif.counts.interval <- function(ma.df, path.to.bigWig, adj, fc, half.win, file.suffix = 'instances.bigWig') {
    decreased =  deseq.df.to.3col.bed(ma.df[ma.df$padj < adj & !is.na(ma.df$padj) & ma.df$log2FoldChange < fc,])
    increased =   deseq.df.to.3col.bed(ma.df[ma.df$padj < adj & !is.na(ma.df$padj) & ma.df$log2FoldChange > fc,])
    not.different =  deseq.df.to.3col.bed(ma.df[ma.df$padj > 0.5 & !is.na(ma.df$padj) & !is.na(ma.df$log2FoldChange) &
                                          abs(ma.df$log2FoldChange) < 0.25,])
    decreased = bed.window(decreased, half.win)
    increased = bed.window(increased, half.win)
    not.different = bed.window(not.different, half.win)

    all.fimo = data.frame(matrix(ncol = 4, nrow = 0))
    colnames(all.fimo) = c('density', 'tf', 'category', 'range')

    for (mod.bigWig in Sys.glob(file.path(path.to.bigWig,
                                          paste('*', file.suffix, sep =''))) {
        factor.name = strsplit(strsplit(mod.bigWig, "/")[[1]][length(strsplit(mod.bigWig, '/')[[1]])], file.suffix)[[1]][1]
        print(factor.name)
        loaded.bw = load.bigWig(mod.bigWig)
        dec.inten = bed.step.probeQuery.bigWig(loaded.bw, decreased,
                                               gap.value = 0, step = 10, as.matrix = TRUE)
        dec.query.df = data.frame(cbind(colMeans(dec.inten), factor.name,
                                        'Decreased', seq(-half.win, half.win, 10)), stringsAsFactors=F)
        colnames(dec.query.df) = c('density', 'tf', 'category', 'range')
```

```
        inc.inten = bed.step.probeQuery.bigWig(loaded.bw, increased,
                                               gap.value = 0, step = 10, as.matrix = TRUE)
        inc.query.df = data.frame(cbind(colMeans(inc.inten), factor.name,
                                        'Increased', seq(-half.win, half.win, 10)), stringsAsFactors=F)
        colnames(inc.query.df) = c('density', 'tf', 'category', 'range')
        ctrl.inten = bed.step.probeQuery.bigWig(loaded.bw,  not.different,
                                                gap.value = 0, step = 10, as.matrix = TRUE)
        ctrl.query.df = data.frame(cbind(colMeans(ctrl.inten), factor.name,
                                         'Unchanged', seq(-half.win, half.win, 10)), stringsAsFactors=F)
        colnames(ctrl.query.df) = c('density', 'tf', 'category', 'range')
        tf.all = rbind(dec.query.df, inc.query.df, ctrl.query.df)
        all.fimo = rbind(all.fimo, tf.all)
    }
    all.fimo[,1] = as.numeric(all.fimo[,1])
    all.fimo[,4] = as.numeric(all.fimo[,4])
    return(all.fimo)
}

#by switching the category and tf variables you can make the panels the TFs and traces categories
#can you make an argument to specificy the switch?

plot.fimo.lattice <- function(dat, fact = 'Motif', summit = 'Hypersensitivity Summit', class= '',
                              num.m = -200, num.p =90, y.low =0, y.high = 0.2,
                              col.lines = c(rgb(0,0,1,1/2), rgb(1,0,0,1/2),
                                            rgb(0.1,0.5,0.05,1/2), rgb(0,0,0,1/2),
                                            rgb(1/2,0,1/2,1/2), rgb(0,1/2,1/2,1/2), rgb(1/2,1/2,0,1/2)),
                              fill.poly = c(rgb(0,0,1,1/4),  rgb(1,0,0,1/4), rgb(0.1,0.5,0.05,1/4),
                                            rgb(0,0,0,1/4), rgb(1/2,0,1/2,1/4))) {
    pdf(paste('composite_', fact, '_signals_', summit, '_peaks', class, '.pdf', sep=''), width=6.83,
        height=3.5)
    print(xyplot(density ~ range|category, groups = tf, data = dat, type = 'l',
                 scales=list(x=list(cex=0.8,relation = "free"), y =list(cex=0.8,
                                                                        #removing this to keep y axes the same
                                                                        #relation="free",
                                                                        axs = 'i')),
                 xlim=c(num.m,num.p),
                 #dynamic y axes
                 #ylim = c(y.low, y.high),
                 col = col.lines,
                 auto.key = list(points=F, lines=T, cex=0.8, columns = 2),
                 par.settings = list(superpose.symbol = list(pch = c(16), col=col.lines, cex =0.7),
                                     superpose.line = list(col = col.lines, lwd=c(2,2,2,2,2,2),
                                                           lty = c(1,1,1,1,1,1,1,1,1,1))),
                 cex.axis=1.0,
                 par.strip.text=list(cex=0.9, font=1, col='black'),
                 aspect=1.0,
                 between=list(y=0.5, x=0.5),
                 index.cond = list(c(3, 2, 1)),
                 lwd=2,
                 ylab = list(label = paste(fact," Density", sep=''), cex =1),
                 xlab = list(label = paste("Distance from ", summit, sep=''), cex =1),
                 #upper = dat$upper,
                 #fill = fill.poly,
                 #lower = dat$lower,
                 strip = function(..., which.panel, bg) {
                     bg.col = c("#2290cf","#ce228e", "grey65")
                 strip.default(..., which.panel = which.panel, bg = rep(bg.col, length = which.panel)[which.panel])
                 },
                 #this commented out is for plotting confidence intervals, (see PRO-seq vignette)
                 #panel = function(x, y, ...){
                     #panel.superpose(x, y, panel.groups = 'my.panel.bands', ...)
                     #panel.xyplot(x, y, ...)
       #}
                 ))
    dev.off()
}


plotting.1h.df = get.motif.counts.interval(df.CAF.sa.1h.deseq,
                              path.to.bigWig = "/Users/guertinlab/Desktop/Dudley_Lab/",
                              adj = 0.0001, fc = 0, half.win = 600, file.suffix = '_1h_mm10_instances.bigWig')
plotting.7d.df = get.motif.counts.interval(df.CAF.sa.7d.deseq,
                              path.to.bigWig = "/Users/guertinlab/Desktop/Dudley_Lab/",
                              adj = 0.0001, fc = 0, half.win = 600, file.suffix = '_7d_mm10_instances.bigWig')

#with more motifs you will need to specify more colors...
plot.fimo.lattice(plotting.1h.df, summit = "Hypersensitivity Summit 1h", num.m = -500, num.p = 500,
                  col.lines = c('#A27E00CC', '#FF4217CC', '#63459aCC',
                    '#6ebe44CC', '#000573CC', '#ff9703CC'))
plot.fimo.lattice(plotting.7d.df, summit = "Hypersensitivity Summit 7d", num.m = -500, num.p = 500,
                  col.lines = c('#A27E00CC', '#FF4217CC', '#63459aCC',
                                '#6ebe44CC', '#000573CC', '#ff9703CC',
                                '#336633', '#FF00CC'))

save(df.CAF, df.CAF.sa.1h.deseq, df.CAF.sa.1h.deseq.grps,
     df.CAF.sa.7d.deseq, df.CAF.sa.7d.deseq.grps,
```

```
    plotting.1h.df, plotting.7d.df,
    file = 'df.macs2.CAF.all.Rdata')
```

The ZNF143 motif is enriched in both increased and decreased classes, this is expected because we observed a variant of the ZNF143 motif as motif 5 in the decreased. You can manually classify and find instances of each motif in the genome (if the MEME e-value is less than 0.01 or some other threshold) and specifically select those with increased or decreased enrichment to plot. For now you can visually look at enrichment, but we can use bootstrapping to estimate a confidence interval and see if they overlap. This bootstrapping code can be lifted from my PRO-seq analysis and modified.

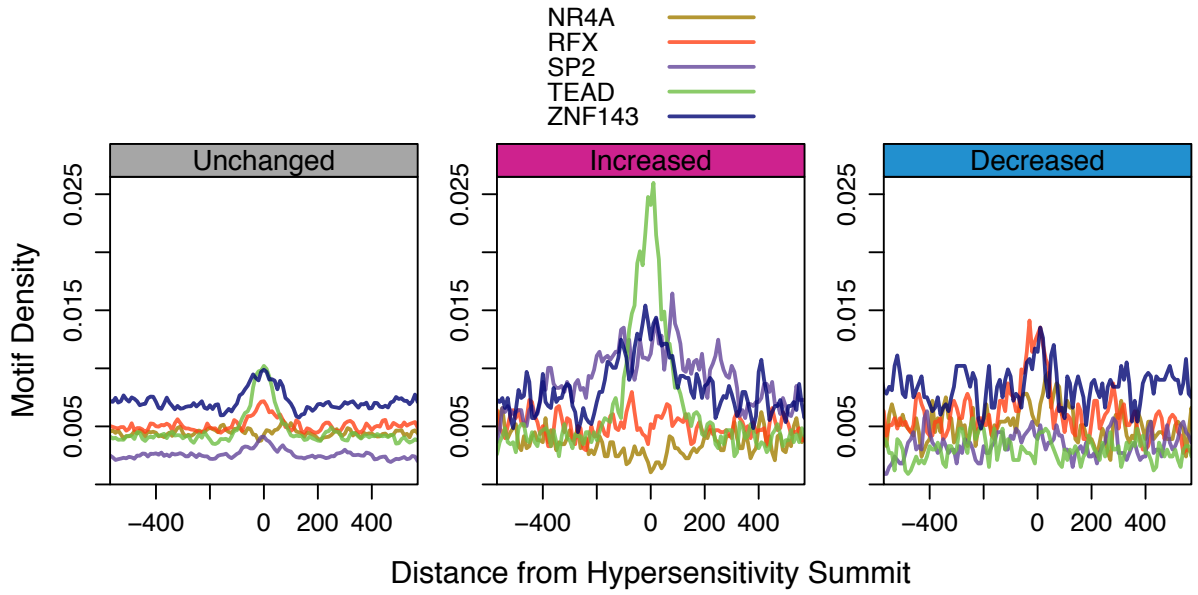


Figure 14: Motif enrichment by change in hypersensitivity class.

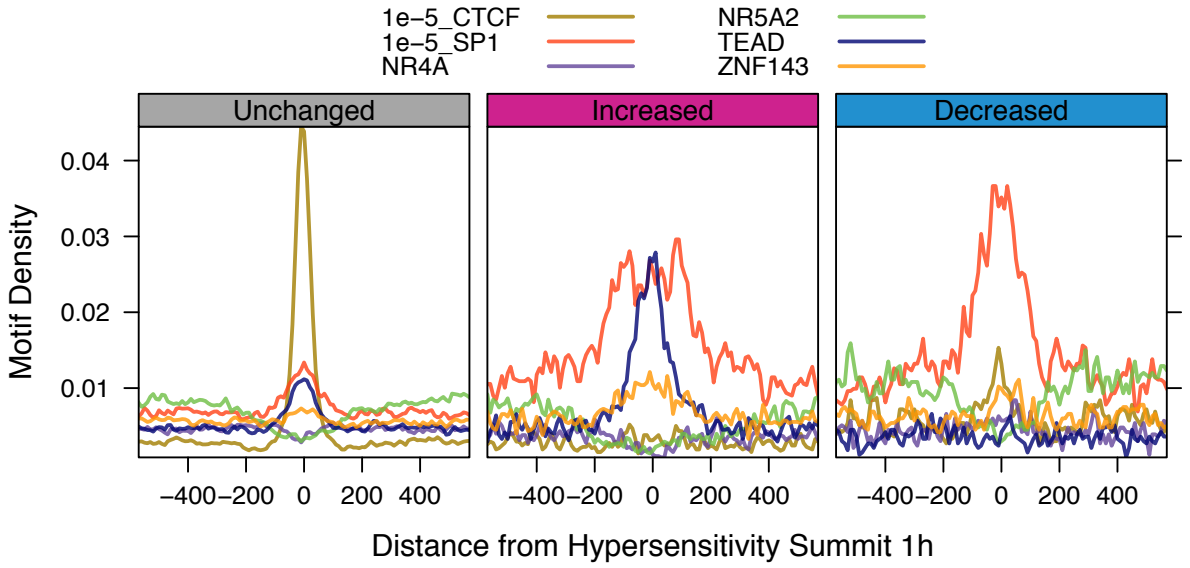


Figure 15: Motif enrichment by change in hypersensitivity class at 1 hour.

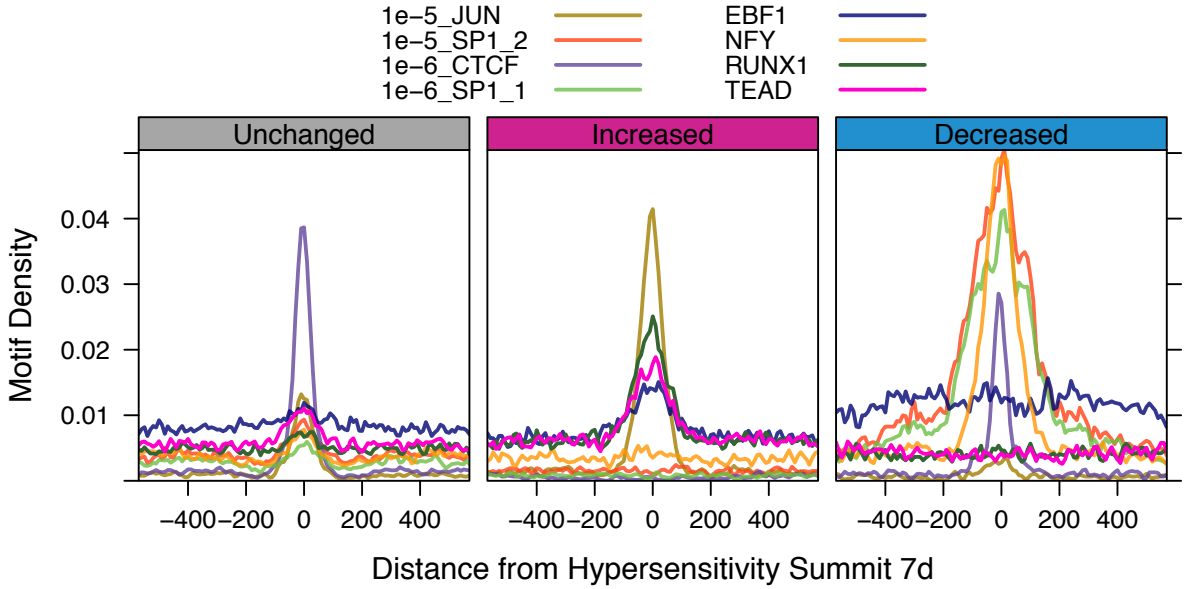


Figure 16: Motif enrichment by change in hypersensitivity class at 7 days.

```
#Comparing percentages of various peak classes with a given motif
PeaksWithMotif <- function(peak.suffix, motif.prefix)
{
    dec = paste("decreased_", peak.suffix, sep = "")
    inc = paste("increased_", peak.suffix, sep = "")
    nodif = paste("notDifferent_", peak.suffix, sep = "")
    motif = paste(motif.prefix, "_mm10_de_novo.fimo/fimo.gff", sep = "")

    nodif.intersect = bedr(input = list(a = nodif, b = motif), method = "intersect", params = c("-wao"))
    nodif.with = sum(as.numeric(nodif.intersect[,ncol]) > 0)
    nodif.without = sum(as.numeric(nodif.intersect[,ncol]) == 0)

    inc.intersect = bedr(input = list(a = inc, b = motif), method = "intersect", params = c("-wao"))
    inc.with = sum(as.numeric(inc.intersect[,ncol]) > 0)
    inc.without = sum(as.numeric(inc.intersect[,ncol]) == 0)

    dec.intersect = bedr(input = list(a = dec, b = motif), method = "intersect", params = c("-wao"))
    ncol = ncol(dec.intersect)
    dec.with = sum(as.numeric(dec.intersect[,ncol]) > 0)
    dec.without = sum(as.numeric(dec.intersect[,ncol]) == 0)

    result = cbind(c(nodif.with, nodif.without),
                   c(inc.with, inc.without),
                   c(dec.with, dec.without))
    colnames(result) <- c("Unchanged", "Increased", "Decreased")
    rownames(result) <- c("with motif", "without motif")

    #Can remove this line if you want the actual counts
    result = 100*sweep(result, 2, colSums(result), "/")

    barplot(result, col = c("red", "darkblue"), legend = rownames(result))
    print(chisq.test(result))
    return(result)
}

#Example with TEAD
library(bedr)
TEAD_1h = PeaksWithMotif("df.macs2.CAF.sa.1h.deseq_1e-04_FDR.bed", "TEAD_1h")
```

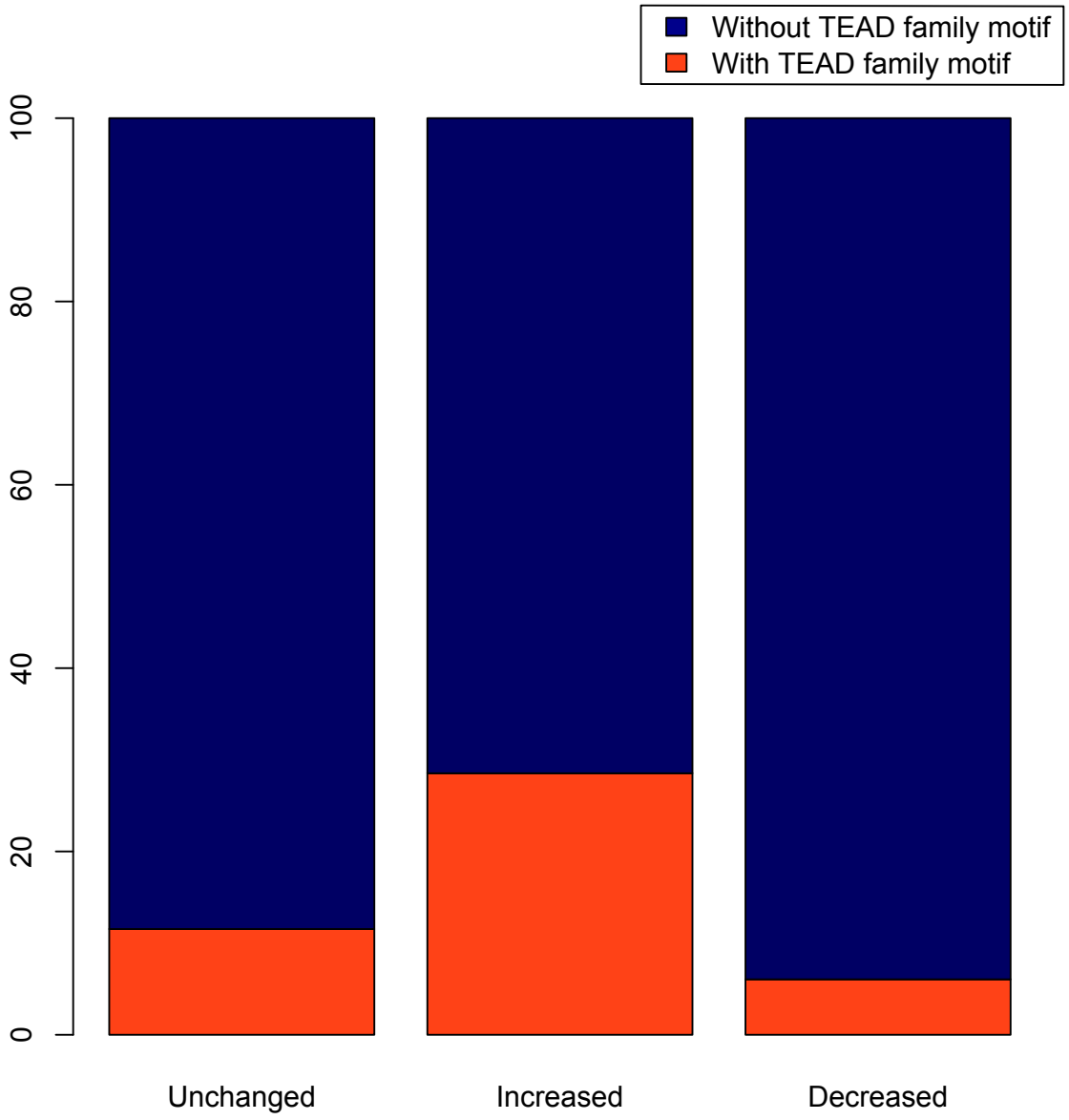


Figure 17: Proportions of peaks with TEAD family motif.

6 Polishing in Adobe Illustrator

I combined the panels in Adobe Illustrator and made some additions.

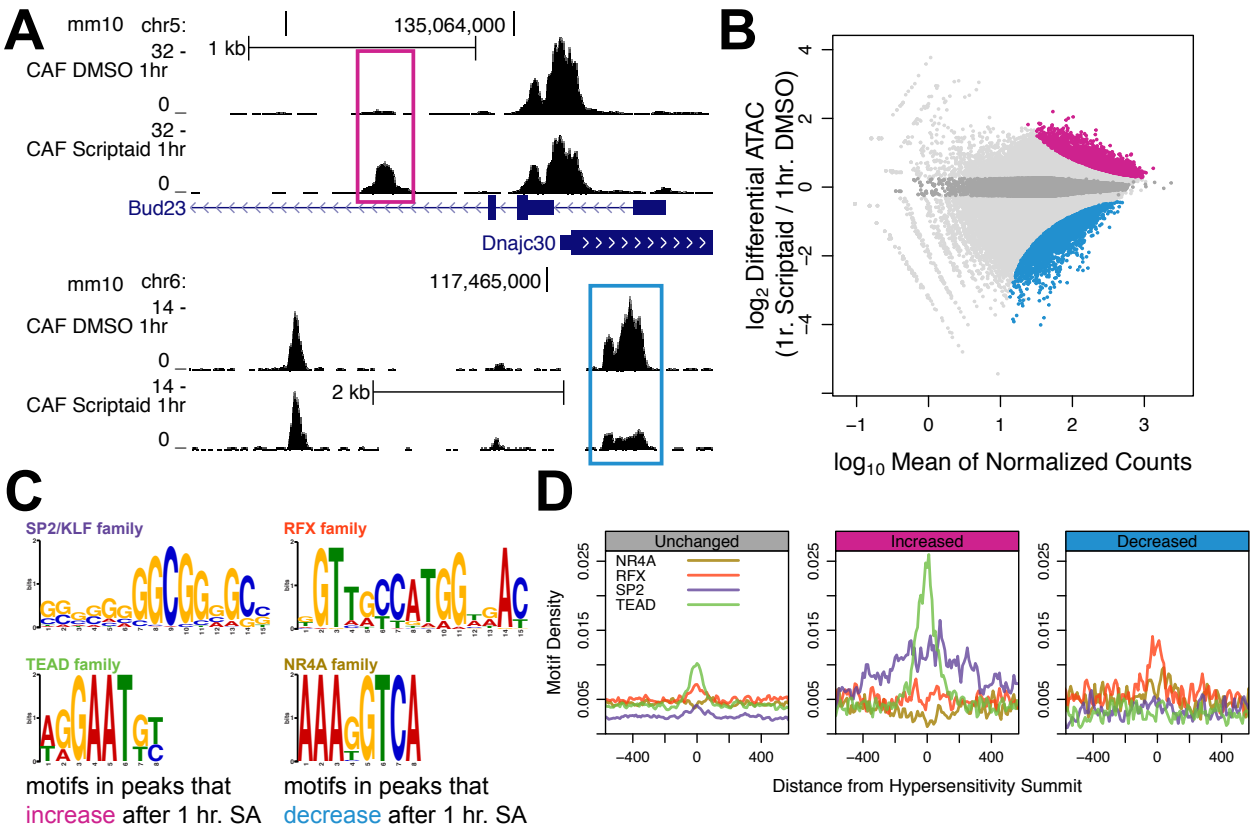


Figure 18: A) ATAC-seq reveals putative regulatory elements in the genome that stay constant, increase, and decrease their accessibility upon Scriptaid treatment. B) We identified 3329 peaks that decrease in chromatin accessibility (blue) and 5774 peaks that increase accessibility (red) after 1 hour Scriptaid treatment. The dark grey points are a conservative set of peaks that do not change upon treatment. C) *de novo* motif analysis identified the SP2 and TEAD transcription factor recognition motifs in the peaks that increase intensity. The RFX and NR4A transcription factor cognate elements were identified *de novo* exclusively in peaks that decrease. D). We identified instances of these motifs in the unchanged, increased, and decreased classes to confirm that motif enrichment is unique to either increased or decreased class and intermediate in the unchanged class.

7 Exploratory rotation project

The goal of the rotation is to introduce you to what we do in the group and learn basics of sequencing analysis. It would be great if you were able to perform some basic comparisons of peak callers and motif analysis software, but I understand that 4 weeks is short and this is all new to you. You should expand upon this vignette below with your figures, thoughts, rationale, etcetera. Also, please correct any errors of mine above. Below are my thoughts:
1) run through the code from above and try to understand the basics and implementation.
2) perform the comparison of 7 day treatment in R.
3) I will get raw data back from the core soon as a part of another collaboration, but it is in human. Performing these same analyses and aligning to hg38 would get you more comfortable with the analysis and you will run into minor issues that you can troubleshoot.
4) Downloading, installing, and running `MACS2` and `F-seq`. I am having issues implementing `hotspot`, so I will need to troubleshoot.
5) Assuming these are running you can call peaks with each and use intersectBed and file line numbers (wc -l) to make a scaled Venn diagram (generally I dislike Venn diagrams, but in this case it coomunicates the point effectively). There may be some nuance you have to consider when using bedtoolsIntersect, you may beed to also use bedtoolsMerge...alternatively you can use R for analysis.
6) once you have lists of coordinates that are unique to specific peak callers you can perform motif analysis on the different categories. If you get known motifs from each category then you can conclude that each peak caller picks up some bona fide transcription factor binding sites. However, if one of the *unique to a particular peak caller* categories just gives back low complexity, repetitive mono/dinucleotide, or no significant motifs, then we can make a tentative conclusion that these are not specific regulatory element peaks. We are always balancing sensitivity and specificity and I would argue that for these exploratory analyses we want sensitivity and we will sacrifice specificity, but this type of analysis will tell us where the additional peaks are increasing sensitivity or just adding false positives. Obviously this is not an exhaustive analysis.
7) Download, install, run and compare different motif callers. I see HOMER used a lot, but I am sure there are dozens that I am not familiar with. A review that has already compared peak callers could be a good start.

Tommy Scott and Arun Dutta

7.1 Comparison of peak callers

```
#First install bedr and dependencies (did manually)
library(bedr)
library(VennDiagram)
directory = "/Users/guertinlab/Desktop/Dudley_Lab/"
setwd(directory)

#Add bedtools to the path
Sys.setenv(PATH = paste(Sys.getenv("PATH"), "/usr/local/bin/bedtools", sep = ":"))

#Takes 2 bed files, intersects them, and makes a venn diagram
VennFromBeds <- function(files, names)
{
    file.1 = files[1]
    file.2 = files[2]
    intersect.1vs2 = bedr(input = list(a = file.1, b = file.2), method = "intersect", params = c("-wao"))
    intersect.2vs1 = bedr(input = list(a = file.2, b = file.1), method = "intersect", params = c("-wao"))
    area1 = nrow(intersect.1vs2)
    area2 = nrow(intersect.2vs1)
    ncol = ncol(intersect.1vs2)
    cross.area = sum(as.numeric(intersect.1vs2[,ncol]) > 0)
    quartz()
    draw.pairwise.venn(area1 = area1, area2 = area2, cross.area = cross.area, category = names,
                       fill = c("red", "blue"), cat.pos = c(-30, 30), ext.dist = rep(0.1, 2))
}

#Example
VennFromBeds(files = c("CAF_macs2_merged_peaks_smallpval.bed", "CAF_hotspot_merged_peaks_smallpval.bed"),
             names = c("MACS2", "Hotspot"))

#Save to pdf
quartz.save(file = 'CAF_macs2_vs_hotspot_merged.pdf', type = "pdf")
```

I did not get a chance to run this because I lost the bigWig files, but by using practice files I think the code works.

```
#Comparing macs2 vs hotspot
for bed in *creased*hotspot*deseq_1e-04*.bed
do
       echo $bed
       frontsame=$(echo $bed | awk -F'.hotspot' '{print $1}')
       backsame=$(echo $bed | awk -F'hotspot.' '{print $2}')
       direction=$(echo $bed | awk -F'_df' '{print $1}')
       time=$(echo $bed | awk -F'.deseq' '{print $1}' | awk -F'.sa.' '{print $2}')

       intersectBed -v -a ${frontsame}.macs2.${backsame} -b ${frontsame}.hotspot.${backsame} \
              > ${direction}_${time}_merged_peaks_smallpval_unique_to_macs2_vs_hotspot.bed
       intersectBed -v -a ${frontsame}.hotspot.${backsame} -b ${frontsame}.macs2.${backsame} \
              > ${direction}_${time}_merged_peaks_smallpval_unique_to_hotspot_vs_macs2.bed
done

for bed in *_merged*unique*.bed
do
       echo $bed
       name=$(echo $bed | awk -F'.bed' '{print $1}')
       fastaFromBed -fi mm10.fa -bed $name.bed -fo $name.fasta
done

for fasta in *1h_merged*unique_to_macs2*.fasta
do
       echo $fasta
       condition=$(echo $fasta | awk -F'_1h_merged' '{print $1}')

       meme-chip -o ${condition}_merged_peaks_smallpval_unique_to_macs2_vs_hotspot.meme_chip_output \
              -dna -meme-nmotifs 15 -meme-minw 5 -meme-maxw 15 \
              -filter-thresh 0.01 -fimo-skip -spamo-skip \
              -neg ${condition}_merged_peaks_smallpval_unique_to_hotspot_vs_macs2.fasta \
              ${condition}_merged_peaks_smallpval_unique_to_macs2_vs_hotspot.fasta
       meme-chip -o ${condition}_merged_peaks_smallpval_unique_to_hotspot_vs_macs2.meme_chip_output \
              -dna -meme-nmotifs 15 -meme-minw 5 -meme-maxw 15 \
              -filter-thresh 0.01 -fimo-skip -spamo-skip \
              -neg ${condition}_merged_peaks_smallpval_unique_to_macs2_vs_hotspot.fasta \
              ${condition}_merged_peaks_smallpval_unique_to_hotspot_vs_macs2.fasta
done
```

7.2 Comparison of MEME 4.12 to MEME 5.0

```
head -n 1000 decreased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.bed > 1000_peaks_dec.bed
head -n 1000 increased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.bed > 1000_peaks_inc.bed

fastaFromBed -fi mm10.fa -bed 1000_peaks_dec.bed -fo 1000_peaks_dec.fasta
fastaFromBed -fi mm10.fa -bed 1000_peaks_inc.bed -fo 1000_peaks_inc.fasta

#Ran this command on my computer (MEME 4.12) and Ashley's computer (MEME 5.0.1)
meme-chip -o 1000_peaks_dec_meme_chip_output -dna -meme-nmotifs 15 -meme-minw 5 -meme-maxw 15 \
       -filter-thresh 0.01 -fimo-skip -spamo-skip -neg 1000_peaks_inc.fasta 1000_peaks_dec.fasta

#Then I realized MEME 4.12 only subsamples 600 peaks (I left out the option telling it to use 3000),
#so I re-ran it on the 600 it subsampled without a negative sample,
#producing almost identical results with both versions
meme-chip -o 600_peaks_dec_meme_chip_output -dna -meme-nmotifs 15 -meme-minw 5 -meme-maxw 15 \
       -filter-thresh 0.01 -fimo-skip -spamo-skip seqs-sampled

#This would be more proper
head -n 600 decreased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.bed > 600_peaks_dec.bed
head -n 600 increased_df.macs2.CAF.sa.1h.deseq_1e-04_FDR.bed > 600_peaks_inc.bed
fastaFromBed -fi mm10.fa -bed 600_peaks_dec.bed -fo 600_peaks_dec.fasta
fastaFromBed -fi mm10.fa -bed 600_peaks_inc.bed -fo 600_peaks_inc.fasta
meme-chip -o 600_peaks_dec_meme_chip_output -dna -meme-nmotifs 15 -meme-minw 5 -meme-maxw 15 \
       -filter-thresh 0.01 -fimo-skip -spamo-skip -neg 600_peaks_inc.fasta 600_peaks_dec.fasta
```

References

Bailey TL, Williams N, Misleh C, Li WW (2006). "MEME: discovering and analyzing DNA and protein sequence motifs." *Nucleic acids research*, **34**(suppl 2), W369–W373.

Buenrostro JD, Giresi PG, Zaba LC, Chang HY, Greenleaf WJ (2013). "Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position." *Nature methods*, **10**(12), 1213–1218.

John S, Sabo PJ, Thurman RE, Sung MH, Biddie SC, Johnson TA, Hager GL, Stamatoyannopoulos JA (2011). "Chromatin accessibility pre-determines glucocorticoid receptor binding patterns." *Nature genetics*, **43**(3), 264.

Karolchik D, Barber GP, Casper J, Clawson H, Cline MS, Diekhans M, Dreszer TR, Fujita PA, Guruvadoo L, Haeussler M, *et al.* (2014). "The UCSC genome browser database: 2014 update." *Nucleic acids research*, **42**(D1), D764–D770.

Langmead B, Trapnell C, Pop M, Salzberg S (2009). "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome." *Genome Biology*, **10**(3), R25. ISSN 1465-6906. doi: 10.1186/gb-2009-10-3-r25. URL http://genomebiology.com/2009/10/3/R25.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, *et al.* (2009). "The sequence alignment/map format and SAMtools." *Bioinformatics*, **25**(16), 2078–2079.

Liu Y, Walavalkar NM, Dozmorov MG, Rich SS, Civelek M, Guertin MJ (2017). "Identification of breast cancer associated variants that modulate transcription factor binding." *PLOS Genetics*, **13**(9), 1–21. doi:10.1371/journal.pgen.1006761. URL https://doi.org/10.1371/journal.pgen.1006761.

Martins AL, Walavalkar NM, Anderson WD, Zang C, Guertin MJ (2018). "Universal correction of enzymatic sequence bias reveals molecular signatures of protein/DNA interactions." *Nucleic acids research*.

Zhang Y, Liu T, Meyer CA, Eeckhoute J, Johnson DS, Bernstein BE, Nusbaum C, Myers RM, Brown M, Li W, *et al.* (2008). "Model-based analysis of ChIP-Seq (MACS)." *Genome Biol*, **9**(9), R137.