A Model to Predict Drivers of Adipogenesis

***Arun B. Dutta*[1], *Bao H. Nguyen*[1], *and Michael J. Guertin*[1]**

[1]University of Virginia, Charlottesville, Virginia

16 October 2020

Contents

1 Background

The following pipeline is designed to compile on the UVA Rivanna Cluster. While most tools and packages are available through Rivanna, you will need to install some manually. To transfer files into and out of Rivanna, use sftp.

2 Retrieving ATAC-seq fastq

3 Processing ATAC-seq reads and aligning with Bowtie2

3.1 Installing fastq_pair

fastq_pair rewrite paired-end fastq files to make sure all reads have a mate, which is required for Bowtie2 to align properly. Remember to move the fastq_pair binary into the $PATH. All files within the /scratch/user directory has a 90 days expiration limit (since last modified) so it is recommended to install packages in a permanent directory like /home/user.

```
mkdir /scratch/bhn9by/ATAC
cd /scratch/bhn9by/ATAC

#Retrieve and build fastq_pair binary
wget https://github.com/linsalrob/fastq-pair/archive/master.zip
unzip master.zip
cd fastq-pair-master
gcc -std=gnu99   main.c robstr.c fastq_pair.c is_gzipped.c  -o fastq_pair
cp fastq_pair /home/bhn9by/bin
cd..
```

3.2 Generate slurm file for each replicate and run in parallel

To run slurm jobs for each replicate in parallel, we concatenate three slurm header files and two lines corresponding to the name of the replicate. Processing and aligning each replicate will take several hours so this slurm_header method will be faster. The content of each header is provided below.

```
#header_1   --> sbatch -n and -t
#temp.txt   --> sbatch -o (name of replicate)
#header_2   --> sbatch -p, -A, and modules to load
#temp2.txt  --> define variable $i as name of replicate
#header_3   --> actual script

for i in *_atac_PE1.fastq.gz
do
    name=$(echo $i | awk -F"/" '{print $NF}' | \
        awk -F"_atac_PE1.fastq.gz" '{print $1}')
    echo $name
    echo '#SBATCH -o' $name'.align.out' > temp.txt
    echo 'i='$i > temp2.txt
    cat align_slurm_header_1.txt temp.txt \
```

```
            align_slurm_header_2.txt temp2.txt \
            align_slurm_header_3.txt > $name.align.slurm
        sbatch $name.align.slurm
        rm temp.txt
        rm temp2.txt
    done
```

3.2.1 align_slurm_header_1.txt

```
    #!/bin/bash
    #SBATCH -n 1
    #SBATCH -t 96:00:00
```

3.2.2 align_slurm_header_2.txt

```
    #SBATCH -p standard
    #SBATCH -A guertinlab

    module load bioconda/py3.6 gcc/7.1.0 bowtie2/2.2.9 samtools/1.10
    source activate myenv
```

3.2.3 align_slurm_header_3.txt

```
    name=$(echo $i | awk -F"/" '{print $NF}' | \
        awk -F"_atac_PE1.fastq.gz" '{print $1}')
    echo $name
    gunzip $name*.gz

    echo 'pairing .fastq files'
    fastq_pair ${name}_atac_PE1.fastq ${name}_atac_PE2.fastq
    rm $name*single.fq
    echo 'align to mouse genome'

    bowtie2 --maxins 500 -x \
            /project/genomes/Mus_musculus/UCSC/mm10/Sequence/Bowtie2Index/genome \
            -1 ${name}_atac_PE1.fastq.paired.fq \
            -2 ${name}_atac_PE2.fastq.paired.fq -S ${name}_atac_smp.sam

    echo 'quality filter and remove duplicate amplicons'
    samtools view -b -q 10 ${name}_atac_smp.sam | samtools sort -n - | \
        samtools fixmate -m - - | samtools sort - | \
        samtools markdup -r - ${name}_atac_rmdup.bam
    rm ${name}_atac_smp.sam
    gzip ${name}*fastq
```

3.3 Simplified version

For clarity we provided a simplified, sequential version of the processing and alignment script.

```
#align .fastq to 10mm genome
for i in *_atac_PE1.fastq.gz
do
    name=$(echo $i | awk -F"/" '{print $NF}' | \
        awk -F"_atac_PE1.fastq.gz" '{print $1}')
    echo $name
    gunzip $name*.gz

    echo 'pairing .fastq files'
    fastq_pair ${name}_atac_PE1.fastq ${name}_atac_PE2.fastq
    rm $name*single.fq

    echo 'align to mouse genome'
    bowtie2 --maxins 500 -x \
        /project/genomes/Mus_musculus/UCSC/mm10/Sequence/Bowtie2Index/genome \
        -1 ${name}_atac_PE1.fastq.paired.fq \
        -2 ${name}_atac_PE2.fastq.paired.fq -S ${name}_atac_smp.sam

    echo 'quality filter and remove duplicate amplicons'
    samtools view -b -q 10 ${name}_atac_smp.sam | samtools sort -n - | \
        samtools fixmate -m - - | samtools sort - | \
        samtools markdup -r - ${name}_atac_rmdup.bam
    rm ${name}_atac_smp.sam
    gzip ${name}*fastq
done
```

4 Performing seqOutBias

seqOutBias corrects for enzymatic sequence bias by scaling the aligned read counts by the ratio of genome-wide observed read counts to the expected sequence based counts for each k-mer. The sequence based k-mer counts take into account mappability at a given read length using Genome Tools' Tallymer program. We will also use seqOutBias to generate bigwig from the bam files.

4.1 Generating slurm file for each replicate

The content of each header is provided below.

```
for bam in *rmdup.bam
do
    name=$(echo $bam | awk -F"_atac_rmdup.bam" '{print $1}')
    echo $name
    echo '#SBATCH -o' $name'.bigwig.out' > temp.txt
    echo 'bam='$bam > temp2.txt
    cat bigwig_slurm_header_1.txt temp.txt \
        bigwig_slurm_header_2.txt temp2.txt \
        bigwig_slurm_header_3.txt > $name.convert.to.bigwig.slurm
done
```

4.1.1 bigwig_slurm_header_1.txt

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 96:00:00
```

4.1.2 bigwig_slurm_header_2.txt

```
#SBATCH -p standard
#SBATCH -A guertinlab

module load genometools/1.5.10 wigtobigwig/2.8 gcc/7.1.0 seqoutbias/1.2.0
```

4.1.3 bigwig_slurm_header_3.txt

```
cd /scratch/bhn9by/ATAC

name=$(echo $bam | awk -F"_atac_rmdup.bam" '{print $1}')
echo $name

seqOutBias \
    /project/genomes/Mus_musculus/UCSC/mm10/Sequence/WholeGenomeFasta/genome.fa \
    $bam --skip-bed --no-scale --bw=${name}.bigWig \
    --only-paired --shift-counts --read-size=38
```

4.2 Running seqOutBias

Run one slurm file to completion to generate requisite tallymer mappability file.

```
sbatch 3T3_20min_rep1.convert.to.bigwig.slurm
```

After this is done, start all others (and repeat the first one).

```
for slurm in *bigwig*slurm
do
    sbatch $slurm
done
```

4.3 Simplified version

For clarity we provided a simplified, sequential version of the seqOutBias script.

```
#perform seqOutBias and convert .bam to .bigwig
cd /scratch/bhn9by/ATAC

for bam in *rmdup.bam
do
    name=$(echo $bam | awk -F"_atac_rmdup.bam" '{print $1}')
    echo $name

    seqOutBias \
    /project/genomes/Mus_musculus/UCSC/mm10/Sequence/WholeGenomeFasta/genome.fa \
         $bam --skip-bed --no-scale --bw=${name}.bigWig \
         --only-paired --shift-counts --read-size=38
done
```

5 Calling ATAC peaks

MACS2 is a program for detecting regions of genomic enrichment. The input is .bam files and the output is .bed files. We will also remove blacklisted regions of the mm10 genome using Bedtools.

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 96:00:00
#SBATCH -o old.peak.calling.out
#SBATCH -p standard
#SBATCH -A guertinlab

module load bioconda/py3.6 gcc/7.1.0 bedtools/2.26.0 samtools/1.10  macs2/2.2.7.1
source activate myenv

#Call peaks
echo 'Calling Peaks'
macs2 callpeak -t *rmdup.bam -f BAMPE -n 3T3_atac \
    --outdir old_peak_calling_macs --keep-dup 50 -q 0.05
```

```
#Download blacklisted mm10 regions
wget https://www.encodeproject.org/files/ENCFF547MET/@@download/ENCFF547MET.bed.gz
gunzip ENCFF547MET.bed.gz
mv ENCFF547MET.bed mm10.blacklist.bed

#Remove blacklisted regions
cd old_peak_calling_macs
bedtools subtract -a 3T3_atac_summits.bed -b ../mm10.blacklist.bed \
    > 3T3_atac_summits_bl_removed.bed
awk '{OFS="\t";} {print $1,$2-99,$3+100,$4,$5}' 3T3_atac_summits_bl_removed.bed \
    > old_peak_calling_summit_window.bed
cd ..
echo 'Done'
```

6 ATAC Preclustering

Process the peaks data through the DESeq2 workflow.

```
library(bigWig)
library(DESeq2)
library(lattice)
library(DEGreport)
source('https://raw.githubusercontent.com/mjg54/znf143_pro_seq_analysis/master/docs/ZNF143_functions.R')

directory = '/scratch/bhn9by/ATAC'
setwd(directory)
preadipo.file = read.table("/scratch/bhn9by/ATAC/old_peak_calling_macs/
    old_peak_calling_summit_window.bed", header = F, sep = "\t")
```

6.1 Assign counts to peaks

A custom function is used to get raw counts from the .bed file and DESeq2 is used to generate a count table for all ATAC peaks.

```
get.raw.counts.interval <- function(df, path.to.bigWig, file.prefix = 'H') {
    df = df[,1:5]
    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[,1:3])
        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)
}

df.preadipo = get.raw.counts.interval(preadipo.file, directory, file.prefix = '3')
save(df.preadipo, file= 'df.preadipo.Rdata')

sample.conditions = factor(sapply(strsplit(as.character(colnames(df.preadipo)), '_'), '[', 2))

sample.conditions = factor(sample.conditions, levels=c("t0","20min","40min",
                                                       "60min","2hr","3hr","4hr"))

deseq.counts.table = DESeqDataSetFromMatrix(df.preadipo,
                as.data.frame(sample.conditions), ~ sample.conditions)

dds = DESeq(deseq.counts.table)

#counts table
normalized.counts.atac = counts(dds, normalized=TRUE)
save(normalized.counts.atac,file='normalized.counts.atac.Rdata')
```

6.2 Plot PCA

Principal component analysis (PCA) deconvolutes the complex data into two dimensions. Replicates of the same treatment should cluster together.

```
#PCA
rld = rlog(dds, blind=TRUE)

x = plotPCA(rld, intgroup="sample.conditions", returnData=TRUE)
plotPCAlattice(x, file = 'PCA_atac.pdf')
```

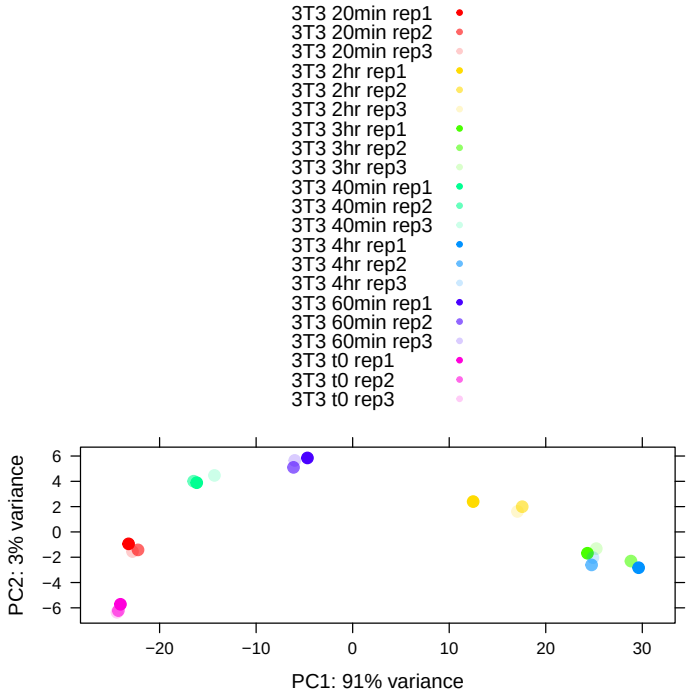

Figure 1: INSERT CAPTION

6.3 Dynamic peaks

Apply the differential functions of DESeq2 to extract dynamic peaks.

```
dds.lrt = DESeq(dds, test="LRT", reduced = ~ 1)

res.lrt = results(dds.lrt)

padj.cutoff = 0.00000001 #1e-8

siglrt.re = res.lrt[res.lrt$padj < padj.cutoff & !is.na(res.lrt$padj),]

rld_mat <- assay(rld)
cluster_rlog = rld_mat[rownames(siglrt.re),]
meta = as.data.frame(sample.conditions)
rownames(meta) = colnames(cluster_rlog)
save(cluster_rlog, meta, sample.conditions, file = 'cluster_rlog_pval_1e8.Rdata')
```

7 ATAC Clustering

Cluster dynamic peaks using degPatterns. Rscript should be run as slurm job due to large memory requirement.

```
library(DESeq2)
library(DEGreport)
library(tibble)
library(lattice)

setwd('/scratch/bhn9by/ATAC')

load('cluster_rlog_pval_1e8.Rdata')

clusters.all.test.1e8 <- degPatterns(cluster_rlog, metadata = meta, minc = 100,
                         time = "sample.conditions", col=NULL, eachStep = TRUE)

save(clusters.all.test.1e8, file = 'clusters.all.minc100.1e8.Rdata')
```

Use the following .slurm script.

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 96:00:00
#SBATCH -o atac.clustering.out
#SBATCH -p largemem
#SBATCH -A guertinlab

module load gcc/7.1.0  openmpi/3.1.4 R/4.0.0

cd /scratch/bhn9by/ATAC

Rscript atac.clustering.R
```

8 ATAC Postclustering

Visualize the clustering with plots of clusters, dendrogram, superclusters. . .

```
library(lattice)
library(data.table)

source('plot.traces.R')

load('clusters.all.minc100.1e8.Rdata')
```

8.1 Generate plot.df object

```
plot.df = clusters.all.test.1e8$normalized

plot.df$sample.conditions = as.character(plot.df$sample.conditions)
plot.df$sample.conditions[plot.df$sample.conditions == 't0'] = 0
plot.df$sample.conditions[plot.df$sample.conditions == '20min'] = 20
plot.df$sample.conditions[plot.df$sample.conditions == '40min'] = 40
plot.df$sample.conditions[plot.df$sample.conditions == '60min'] = 60
```

```
plot.df$sample.conditions[plot.df$sample.conditions == '2hr'] = 120
plot.df$sample.conditions[plot.df$sample.conditions == '3hr'] = 180
plot.df$sample.conditions[plot.df$sample.conditions == '4hr'] = 240
plot.df$sample.conditions = as.numeric(plot.df$sample.conditions)
plot.df = plot.df[order(plot.df$genes),]
plot.df = plot.df[order(plot.df$sample.conditions),]

plot.df$cluster = paste('cluster', as.character(plot.df$cluster), sep = '')

plot.df$chr = sapply(strsplit(plot.df$genes, '[.]'), '[', 1)
plot.df$start = sapply(strsplit(plot.df$genes, '[.]'), '[', 2)
plot.df$end = sapply(strsplit(plot.df$genes, '[.]'), '[', 3)

save(plot.df,file='plot.df.Rdata')

write.table(cbind(plot.df$chr, plot.df$start, plot.df$end),
            file = 'dynamic_peaks.bed', quote = FALSE,
            sep = '\t', col.names=FALSE, row.names=FALSE)
```

8.2 Plot all clusters

```
for (i in unique(plot.df$cluster)) {
    print(i)
    write.table(plot.df[plot.df$cluster == i,
                        c('chr','start','end', 'value', 'cluster')][
                            !duplicated(plot.df[plot.df$cluster == i,]$genes),],
                file = paste0('cluster_bed_',
                              gsub(" ", "", i, fixed = TRUE),'.bed'),
                quote = FALSE, row.names = FALSE, col.names = FALSE, sep = '\t')
}

pdf('atac_clusters.pdf', width=11, height=15)

trellis.par.set(box.umbrella = list(lty = 1, col="black", lwd=1),
                box.rectangle = list( lwd=1.0, col="black", alpha = 1.0),
                plot.symbol = list(col="black", lwd=1.0, pch ='.'))
print(
xyplot(value ~  sample.conditions | cluster, group = genes, data = plot.df,
       type = c('l'),#type = c('l','p'),
       scales=list(x=list(cex=1.0,relation = "free", rot = 45),
       y =list(cex=1.0, relation="free")),
       aspect=1.0,
       between=list(y=0.5, x=0.5),
       ylab = list(label = 'Normalized ATAC signal', cex =1.0),
       xlab = list(label = 'Time (minutes)', cex =1.0),
       par.settings = list(superpose.symbol = list(pch = c(16),
                                                   col=c('grey20'), cex =0.5),
                           strip.background=list(col="grey80"),
                           superpose.line = list(col = c('#99999980'), lwd=c(1),
```

```
                                                    lty = c(1))),
            panel = function(x, y, ...) {
                panel.xyplot(x, y, ...)
                panel.bwplot(x, y, pch = '|', horizontal = FALSE, box.width = 15,
                             do.out = FALSE)
                panel.spline(x, y, col = 'blue', lwd =2.0, ...)
        })

              )
        dev.off()

        #up.flat - 9,23,17,11
        #grad.up - 5,8,10,1
        #up.down - 6,2,12
        #grad.down - 7,3,4,13,14
        #down.up - 24
```

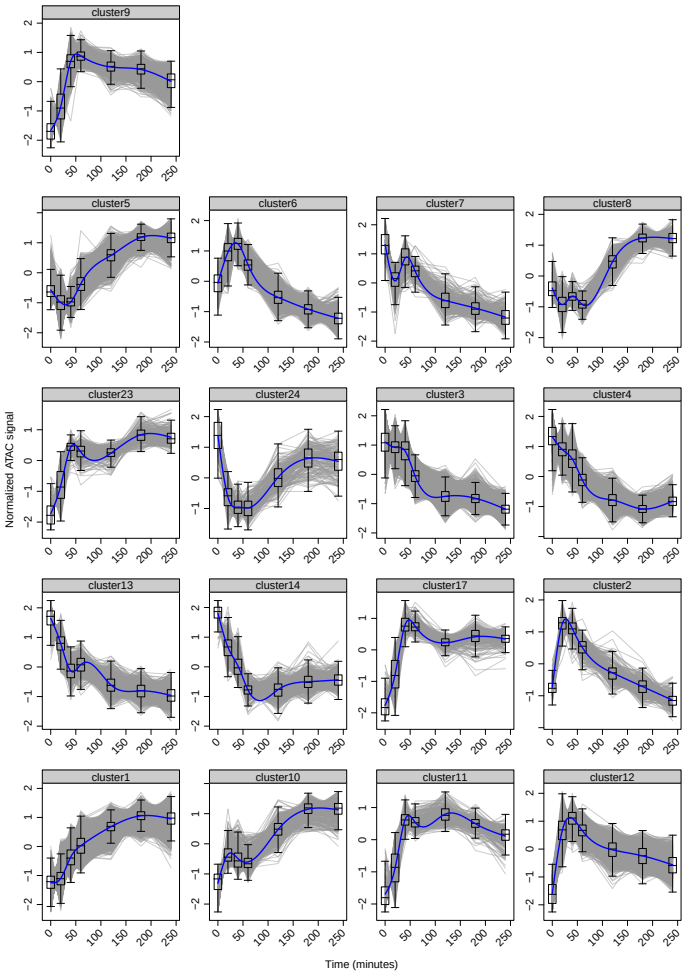

Figure 2: INSERT CAPTION

8.3 Plot dendrogram

```
x = as.data.table(plot.df)
plot.df.cluster = dcast(x, genes + cluster ~ sample.conditions, value.var="value")

avg.clusters = as.data.frame(matrix(nrow = 0, ncol = 7))
for (i in unique(plot.df.cluster$cluster)) {
    z = data.frame(matrix(colMeans(plot.df.cluster[plot.df.cluster$cluster == i,3:9]),
                          ncol = 7, nrow = 1))
    rownames(z) = c(i)
    colnames(z) = as.character(colnames(plot.df.cluster)[3:9])
    avg.clusters = rbind(avg.clusters, z)
}


dd = dist(avg.clusters)
hc = hclust(dd, method = "complete")

pdf('dendrogram.pdf', width=8, height=5)
plot(hc, xlab = "Clusters", main = ' ', hang = -1)
abline(h = 2, lty = 2)
dev.off()
```

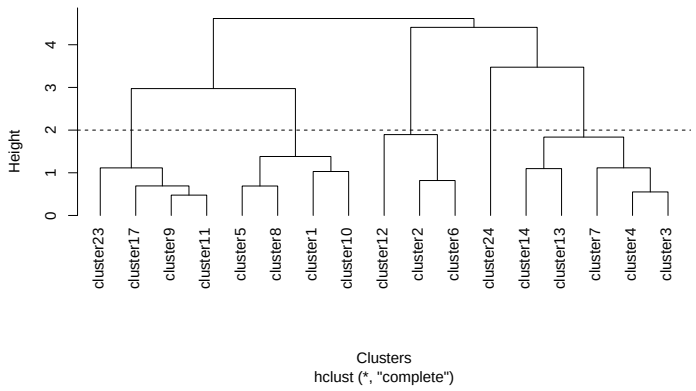

Figure 3: INSERT CAPTION

8.4 Plot clusters organized by supercluster

```
df = data.frame(index=1:17,cluster=unique(plot.df$cluster)[order(unique(plot.df$cluster))])
df$cluster.num = as.integer(sapply(strsplit(df$cluster, 'cluster'), '[', 2))
df = df[order(df$cluster.num),]
df = df[reorder(df$cluster.num,c(23,17,9,11,5,8,1,10,12,2,6,24,14,13,7,4,3)),]

pdf('atac_clusters_org_by_sc.pdf', width=11, height=15)

trellis.par.set(box.umbrella = list(lty = 1, col="black", lwd=1),
```

```
                    box.rectangle = list( lwd=1.0, col="black", alpha = 1.0),
                    plot.symbol = list(col="black", lwd=1.0, pch ='.'))
        print(
            xyplot(value ~  sample.conditions | cluster, group = genes, data = plot.df,
               type = c('l'),#type = c('l','p'),
               scales=list(x=list(cex=1.0,relation = "free", rot = 45),
               y =list(cex=1.0, relation="free")),
               aspect=1.0,
               layout = c(5,5),
               between=list(y=0.5, x=0.5),
               index.cond=list(rev(df$index)),
               skip = c(F,F,F,F,F,
                        F,T,T,T,T,
                        F,F,F,T,T,
                        F,F,F,F,T,
                        F,F,F,F,T),
               ylab = list(label = 'Normalized ATAC signal', cex =1.0),
               xlab = list(label = 'Time (minutes)', cex =1.0),
               par.settings = list(superpose.symbol = list(pch = c(16),
                                                           col=c('grey20'), cex =0.5),
                                   strip.background=list(col="grey80"),
                                   superpose.line = list(col = c('#99999980'), lwd=c(1),
                                                         lty = c(1))),
               panel = function(x, y, ...) {
                   panel.xyplot(x, y, ...)
                   panel.bwplot(x, y, pch = '|', horizontal = FALSE, box.width = 15,
                                do.out = FALSE)
                   panel.spline(x, y, col = 'blue', lwd =2.0, ...)
        })

              )
        dev.off()
```

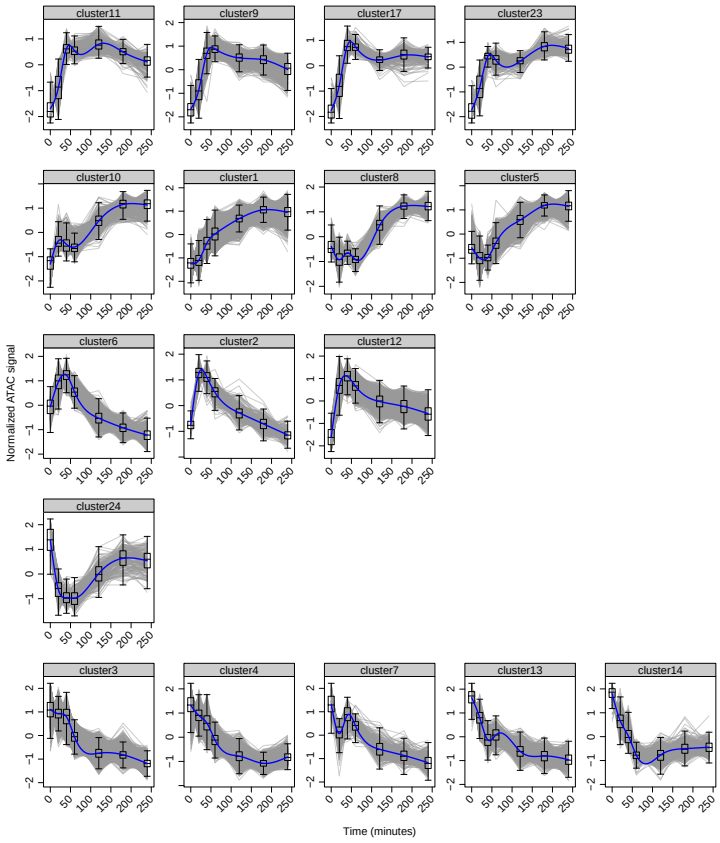

Figure 4: INSERT CAPTION

8.5 Plot supercluster traces

```
gradual.down = plot.df[plot.df$cluster == 'cluster7' |
                       plot.df$cluster == 'cluster3' |
                       plot.df$cluster == 'cluster4' |
                       plot.df$cluster == 'cluster13' |
                       plot.df$cluster == 'cluster14',]

down.up = plot.df[plot.df$cluster == 'cluster24',]

gradual.up = plot.df[plot.df$cluster == 'cluster5' |
                       plot.df$cluster == 'cluster8' |
                       plot.df$cluster == 'cluster10' |
                       plot.df$cluster == 'cluster1',]

up.flat = plot.df[plot.df$cluster == 'cluster9' |
```

```
                        plot.df$cluster == 'cluster23' |
                        plot.df$cluster == 'cluster17' |
                        plot.df$cluster == 'cluster11',]

up.down = plot.df[plot.df$cluster == 'cluster6' |
                        plot.df$cluster == 'cluster12' |
                        plot.df$cluster == 'cluster2',]

gradual.down$supercluster = 'gradual.down'
down.up$supercluster = 'down.up'
gradual.up$supercluster = 'gradual.up'
up.flat$supercluster = 'up.flat'
up.down$supercluster = 'up.down'

nrow(gradual.down)/7
nrow(down.up)/7
nrow(gradual.up)/7
nrow(up.flat)/7
nrow(up.down)/7

plot.df.atac = rbind(gradual.down,
      down.up,
      gradual.up,
      up.flat,
      up.down)
plot.df.atac = plot.df.atac[,-(7:26)]
plot.df.atac$genes = paste0(plot.df.atac$chr,':',plot.df.atac$start,'-',plot.df.atac$end)
save(plot.df.atac,file='plot.df.atac.Rdata')

pdf('atac_superclusters.pdf', width=6.83, height=5)

trellis.par.set(box.umbrella = list(lty = 1, col="black", lwd=1),
                box.rectangle = list( lwd=1.0, col="black", alpha = 1.0),
                plot.symbol = list(col="black", lwd=1.0, pch ='.'))
print(
xyplot(value ~  sample.conditions | supercluster, group = genes,
       data = plot.df.atac, type = c('l'),#type = c('l','p'),
       scales=list(x=list(cex=1.0,relation = "free", rot = 45),
       y =list(cex=1.0, relation="free")),
       aspect=1.0,
       between=list(y=0.5, x=0.5),
       layout = c(5,1),
       ylab = list(label = 'Normalized ATAC signal', cex =1.0),
       xlab = list(label = 'Time (minutes)', cex =1.0),
       par.settings = list(superpose.symbol = list(pch = c(16),
                                                   col=c('grey20'), cex =0.5),
                           strip.background=list(col="grey80"),
                           superpose.line = list(col = c('#99999980'), lwd=c(1),
                                                 lty = c(1))),
       panel = function(x, y, ...) {
           panel.xyplot(x, y, ...)
```

```
                panel.bwplot(x, y, pch = '|', horizontal = FALSE, box.width = 15,
                             do.out = FALSE)
                panel.spline(x, y, col = 'blue', lwd =2.0, ...)

        })

              )
        dev.off()
```

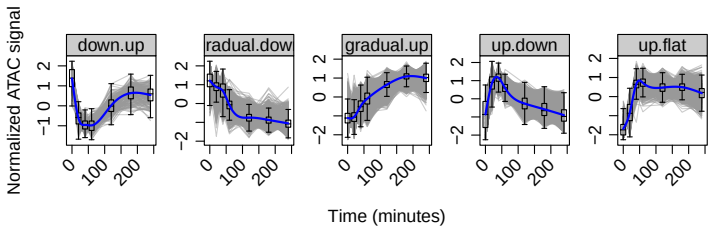

Figure 5: INSERT CAPTION

8.6 Plot individual traces

```
        df = data.frame(sc = c("down.up","gradual.down","gradual.up","up.down","up.flat"),
                        col = c('orange1','#4169E1','#FF0000','green2','#A020F0'))

        for(sc in unique(plot.df.atac$supercluster)) {
            col = df[df$sc == sc,]$col

            y = plot.df.atac[plot.df.atac$supercluster == sc,]

            pdf(file=paste0(sc,'.traces.pdf'))

            trellis.par.set(box.umbrella = list(lty = 1, col="black", lwd=1),
                            box.rectangle = list( lwd=1.0, col="black", alpha = 1.0),
                            plot.symbol = list(col="black", lwd=1.0, pch ='.'))

            print(
                xyplot(value ~  sample.conditions, group = genes, data = y, type = c('l'),
                       scales=list(x=list(cex=1.0,relation = "free", rot = 45),
                       y = list(cex=1.0, relation="free")),
                       aspect=1.0,
```

```
                    between=list(y=0.5, x=0.5),
                    main = list(label = paste0(sc, ' traces'), cex = 1.5),
                    ylab = list(label = 'Normalized ATAC signal', cex =1.0),
                    xlab = list(label = 'Time (minutes)', cex =1.0),
                    par.settings = list(superpose.symbol = list(pch = c(16),
                                        col=c('grey20'), cex =0.5),
                                        strip.background = list(col="grey80"),
                                        superpose.line = list(col = c('#99999980'),
                                        lwd=c(1),lty = c(1))),
                    panel = function(x, y, ...) {
                        panel.xyplot(x, y, ...)
                        panel.bwplot(x, y, pch = '|', horizontal = FALSE, box.width = 15,
                                     do.out = FALSE)
                        panel.spline(x, y, col = 'blue', lwd = 3.5, ...)
                        #replace col = 'blue' with col = col for different sc colors
                    })
        )

        dev.off()

    }
```

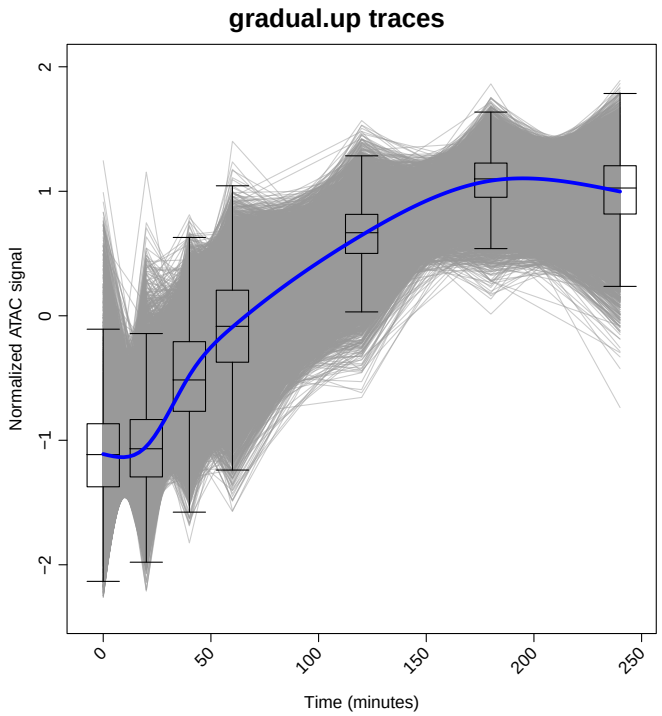

Figure 6: INSERT CAPTION

8.7 Generating .bed file for each cluster

We will generate .bed file for each cluster in order to perform motif enrichment analyses.

```
for (i in unique(plot.df.atac$supercluster)) {
    print(i)
    write.table(plot.df.atac[plot.df.atac$supercluster == i,
                  c('chr','start','end', 'value', 'supercluster')][
                  !duplicated(plot.df.atac[plot.df.atac$supercluster == i,]$genes),],
                file = paste0(i,'.bed'),
                quote = FALSE, row.names = FALSE, col.names = FALSE, sep = '\t')
}
```

9 FIMO motif enrichment

9.1 Find motifs enriched in cluster

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -t 96:00:00
#SBATCH -o fimo.enrichment.out
#SBATCH -p standard
#SBATCH -A guertinlab

module load gcc/7.1.0 bedtools/2.26.0 openmpi/3.1.4 R/4.0.0

cd /scratch/bhn9by/ATAC

#collect ATAC peaks that were not sorted into clusters
#this will serve as a nondynamic control set of regions
intersectBed -v -a old_peak_calling_macs/old_peak_calling_summit_window.bed \
                -b cluster*bed > nondynamic_peaks.bed

tab=$'\t'

mkdir fimo_motif_enrichment
cd fimo_motif_enrichment

#loop through each motif
for motif in /scratch/bhn9by/ATAC/Top_motif/*bed
do
    motif_name=$(echo $motif | awk -F"/" '{print $NF}' | awk -F".bed" '{print $1}')
    echo $motif_name
    touch ${motif_name}.txt
    echo "name""$tab""with.motif""$tab""without.motif" >> ${motif_name}.txt

    #find how many nondynamic peaks contain the motif
    nondyn_with_motif=$(intersectBed -wa -a ../nondynamic_peaks.bed -b $motif | wc -l)
    nondyn_without_motif=$(intersectBed -v -a ../nondynamic_peaks.bed -b $motif | wc -l)

    echo "nondynamic$tab$nondyn_with_motif$tab$nondyn_without_motif" >> ${motif_name}.txt

    #loop though each cluster
```

```
    for bed in ../cluster*bed
    do
    name=$(echo $bed | awk -F"cluster_bed_" '{print $2}' | awk -F".bed" '{print $1}')
    #find how many peaks in each cluster contain the motif

    #calculate percentage of cluster peaks that have the motif
    cluster_with_motif=$(intersectBed -wa -a $bed -b $motif | wc -l)
    cluster_without_motif=$(intersectBed -v -a $bed -b $motif | wc -l)

    echo "$name$tab$cluster_with_motif$tab$cluster_without_motif" >> ${motif_name}.txt

    done

    Rscript ../fimo_motif_enrichment.R ${motif_name}.txt

done
cd ..
```

9.2 fimo_motif_enrichment.R

Use the following Rscript for the loop above.

```
Args=commandArgs(TRUE)
motif = Args[1]

library(lattice)

setwd('/scratch/bhn9by/ATAC/fimo_motif_enrichment')

supercluster.key = data.frame(row.names = c(
                        'cluster9','cluster23','cluster17','cluster11',
                        'cluster5','cluster8','cluster10','cluster1',
                        'cluster6','cluster2','cluster12',
                        'cluster7','cluster3','cluster4','cluster13','cluster14',
                        'cluster24',
                        'nondynamic'),
                        supercluster = c(rep('up.flat',4),
                                         rep('grad.up',4),
                                         rep('up.down',3),
                                         rep('grad.down',5),
                                         rep('down.up',1),
                                         'na')
                        )

colors = c('#000000','#dddddd')

motif.name = strsplit(motif,'.txt')[[1]][1]
print(motif.name)
table = t(read.table(motif,sep='\t',header=T,row.names=1))
```

```
result = 100*sweep(table, 2, colSums(table), "/")

result = result[,c('cluster9','cluster23','cluster17','cluster11',
                   'cluster5','cluster8','cluster10','cluster1',
                   'cluster6','cluster2','cluster12',
                   'cluster7','cluster3','cluster4','cluster13','cluster14',
                   'cluster24',
                   'nondynamic')]

sig = FALSE
sig.clusters = c()
for (cluster in colnames(result)) {
    small.table = result[,c(cluster,'nondynamic')]
    output = chisq.test(small.table)
    if (output$p.value < 0.01) {

        change = ''
        if (small.table[1,2] < small.table[1,1]) {
            change = 'enriched'
            } else {
                change = 'depleted'
            }

        sc = as.character(supercluster.key[cluster,])

        write(paste0(motif.name,'\t',cluster,'\t',change,'\t',sc),
                file="significant_motifs.txt",append=TRUE)
        sig=TRUE
    }
    if (sig == TRUE) {
        pdf(file = paste0(motif.name,'.enrichment.barchart.pdf'),height=9)
        par(las=2)
        barplot(result, col = colors, cex.names= 1.2, legend.text = TRUE,
                args.legend = list(x=2.5,y=113), main = paste0(motif.name,' Enrichment'))
        dev.off()
    }
}
```

10 MEME motif enrichment

11 MEME-FIMO analysis

12 Defining motif families and generating composites