File Processing and Scripts, Related to STAR Methods

Kumagai and Dunphy (2020)

File processing

Trim Fastq files.

```
trim_galore --fastqc --paired -q 20 *.fq.gz
```

Align with human index and yeast index.

```
sample=WT_rep1 &&
bowtie2 -p 16 -x GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.bowtie_index \
-1 "${sample}"*val_1.fq.gz -2 "${sample}"*val_2.fq.gz \
-S /sam_files/"${sample}".sam --local --very-sensitive-local --no-unal --no-mixed \
--no-discordant --phred33 -I 10;

bowtie2 -p 16 -x /Bowtie2Index_yeast/genome -1  "${sample}"*val_1.fq.gz \
-2 "${sample}"*val_2.fq.gz -S /spike_in_yeast/"${sample}".sam --local \
--very-sensitive-local --no-unal --no-mixed --no-discordant --phred33 --no-overlap \
--no-dovetail -I 10
```

Remove mtDNA etc. and make bam files and index file.

```
for file in *.sam; do sed '/chrM/d;/random/d;/chrUn/d' ${file} | samtools view -b \
-q 30  | samtools sort -o /bam_files/${file/%.sam/.bam}
# Make index.
for file in *.bam; do samtools index ${file}
```

Count mapped reads.

```
samtools view -F 0x4 WT_rep1.bam | cut -f 1 | sort | uniq | wc -l
```

BigWig files for MTBP-WT, MTBP-deltaC, and control

Blacklist file was obtained from
http://mitra.stanford.edu/kundaje/akundaje/release/blacklists/.
Spike-in-ratios (number of reads aligned to human genome)/(number of reads aligned to yeast genome) were calculated for each experiment. Scale factors were calculated by using spike-in-ratio (control or MTBP-ΔC)/spike-in-ratio (MTBP-WT) to normalize the reads to MTBP-WT. Scale factors ranged between 0.82 and 1.09.
WT.bw, deltaC.bw, and cont.bw files were made as follows.

```
bamCoverage -b WT_rep1.bam -o WT_rep1.bw -bs 10 --normalizeUsing RPGC \
--ignoreDuplicates --effectiveGenomeSize 2913022398 --scaleFactor 1 \
--blackListFileName  /refs/hg38_blacklist.bed
## Do the same for each file.  Use own scaleFactors.

# Merge three files for WT, deltaC, and control.
bigWigMerge WT_rep1.bw WT_rep2.bw WT_rep3.bw WT.bedgraph

# Sort and get average scores.
sort -k1,1 -k2,2n WT.bedgraph | \
awk 'BEGIN{OFS = "\t"}($5 = $4/3){print $1,$2,$3,$5}' > WT_adjusted.bedgraph

# Make bw file.
bedGraphToBigWig WT_adjusted.bedgraph /refs/hg38.chrom.sizes WT.bw

# BigWig files for H3K4me2 and control-IgG were made in a similar manner.
```

Peak calling with MACS2 for MTBP-WT, MTBP-deltaC, and control

```
# Call peaks with MACS2 for each replicate.
callpeak -t WT_rep1.bam -c cont_rep1.bam -f BAMPE --nolambda -g 3.05e9 -q 0.01 \
--call-summits -n WT --outdir rep1

callpeak -t deltaC_rep1.bam -c cont_rep1.bam -f BAMPE --nolambda -g 3.05e9 -q 0.01 \
--call-summits -n deltaC --outdir rep1

# Obtain intersection of the replicates. Figure S2.
intervene venn -i rep1/WT_peaks.narrowPeak rep2/WT_peaks.narrowPeak \
rep3/WT_peaks.narrowPeak --save-overlaps -o intervene_three_reps/WT \
--names=rep1, rep2, rep3

intervene venn -i rep1/deltaC_peaks.narrowPeak rep2/deltaC_peaks.narrowPeak \
rep3/deltaC_peaks.narrowPeak --save-overlaps -o intervene_three_reps/deltaC \
--names=rep1, rep2, rep3

# Concatenate the peaks that are in overlapped regions.
# Peaks that are common to at least two out of three replicates are selected.
cat 011_rep2_rep3.bed 101_rep1_rep3.bed 110_rep1_rep2.bed 111_rep1_rep2_rep3.bed \
> WT_rep.bed

cat 011_rep2_rep3.bed 101_rep1_rep3.bed 110_rep1_rep2.bed 111_rep1_rep2_rep3.bed \
> deltaC_rep.bed

# Call peaks again with three pooled replicates.
callpeak -t WT_rep1.bam WT_rep2.bam WT_rep3.bam -c cont_rep1.bam cont_rep2.bam \
cont_rep3.bam --nolambda -f BAMPE -g 3.05e9 -q 0.01 -n WT --outdir MACS2/WT_pooled

callpeak -t deltaC_rep1.bam deltaC_rep2.bam deltaC_rep3.bam -c cont_rep1.bam \
cont_rep2.bam cont_rep3.bam --nolambda -f BAMPE -g 3.05e9 -q 0.01 -n WT \
--outdir MACS2/deltaC_pooled

# Call peaks for control files.
```

```
callpeak -t cont_rep1.bam cont_rep2.bam cont_rep3.bam -c cont_rep1.bam \
cont_rep2.bam cont_rep3.bam --nolambda -f BAMPE -g 3.05e9 -q 0.01 -n WT \
--outdir MACS2/cont_pooled
```

Intersection of pooled peaks and replicated peaks was selected and peaks in blacklisted area and peaks that intersect with large control peaks were removed to yield WT.bed and deltaC.bed.

```
bedtools intersect -a WT_pooled/WT_peaks.narrowPeak \
-b intervene_three_reps/WT/sets/WT_rep.bed -wa -u > WT_rep.bed
# Select all the replicated peaks (29044 peaks) out of 64528 peaks.

bedtools intersect -a WT_rep.bed -b hg38_blacklist.bed -v > WT_noblack.bed
# 29038 peaks

bedtools intersect -a WT_noblack.bed -b cont_WT.bed -v > WT.bed
# Do the same for deltaC.
bedtools intersect -a deltaC_noblack.bed -b cont_deltaC.bed -v > deltaC.bed
```

Peak calling with MACS2 for H3K4me2 and control IgG was performed without –nolambda. Two replicates for each sample were used and reproducible peaks between two replicates were selected.

Figures 2 and S2

Figure S2B

Selection of reproducible peaks.

```
library(ggpubr)
library(dplyr)
library(cowplot)
WT_all <- read.delim("MACS2/WT_pooled/WT_peaks.narrowPeak", header = F)
WT_selected <- read.delim("bed_files/WT.bed", header = F)
WT <- bind_rows("all"=WT_all, "selected"=WT_selected, .id="WT")
Q_6_WT <- WT_selected %>% filter(V9 > 6) # Select peaks that have the -log10qvalue of
> 6.
wt_q <- nrow(Q_6_WT)/nrow(WT_selected) # 99.7% of the peaks have -log10qvalue of > 6.
FigS2B_WT <- ggdensity(WT, x = "V9", color = "WT", fill = "WT", palette = c("#00AFBB"
,"#E7B800"), xlim = c(0, 80), ylim = c(0, 5200), y = "..count..", xlab = "-log10qvalu
e")+
  theme(legend.title = element_blank(), legend.position = "none", plot.title = elemen
t_text(hjust = 0.5)) + ggtitle("WT")

# Do the same for deltaC.
deltaC_all <- read.delim("MACS2/deltaC_pooled/deltaC_peaks.narrowPeak", header = F)
deltaC_selected <- read.delim("bed_files/deltaC.bed", header = F)
deltaC <- bind_rows("all"=deltaC_all, "selected"=deltaC_selected, .id="deltaC")
Q_6_deltaC <- deltaC_selected %>% filter(V9 > 6) # Select peaks that have the -log10q
value of > 6.
deltaC_q <- nrow(Q_6_deltaC)/nrow(deltaC_selected) # 99.0% of the peaks have -log10qv
alue of > 6.
```

```
FigS2B_deltaC <- ggdensity(deltaC, x = "V9", color = "deltaC", fill = "deltaC", palet
te = c("#00AFBB","#E7B800"), xlim = c(0, 80), ylim = c(0, 5200), y = "..count..",
                           xlab = "-log10qvalue")+
  theme(legend.title = element_blank(), legend.position = c(0.5, 0.8), legend.text =
element_text(size = 12), plot.title = element_text(hjust = 0.5)) + ggtitle("deltaC")

plot_grid(FigS2B_WT, FigS2B_deltaC)
wt_q
deltaC_q
```

Figure 2B

Combine WT.bed and deltaC.bed.

```
intervene venn -i WT.bed deltaC.bed --save-overlaps -o intervene/WT_deltaC
# WT_only, WT_deltaC, and deltaC_only files are in intervene/WT_deltaC/sets folder.
```

Finding peak summits

Peaks were called again with –call-summits in MACS2 as above.

```
# Intersect final peak files (WT.bed and deltaC.bed) and MACS2 peak files with --call
-summits to obtain summits.
# Many peaks have multiple summits.
bedtools intersect -a bed_files/WT.bed -b MACS2/WT_summits/WT_summits.bed -wa -wb \
> bed_files/WTwithsummits.bed;
bedtools intersect -a bed_files/intervene/WT_deltaC/sets/01_deltaC.bed \
-b MACS2/deltaC_summits/deltaC_summits.bed -wa -wb \
> bed_files/deltaCwithsummits.bed
# deltaC only peaks.
```

Make bed files with peaks containing location information of the summits.

```
# Group by the same ID and take top scored row in the group.
# Then pick one of the rows for the summits with same value.
WTwithsummits <- read.delim("bed_files/WTwithsummits.bed", header = F) %>%
    group_by(V4) %>% top_n(1, V15) %>% distinct(V4, .keep_all = TRUE) %>%
    select(V11,V12,V13,V4)

deltaCwithsummits <- read.delim("bed_files/deltaCwithsummits.bed", header = F) %>%
    group_by(V4) %>% top_n(1, V15) %>% distinct(V4, .keep_all = TRUE) %>%
    select(V11,V12,V13,V4)

MTBP_summits <- bind_rows(WTwithsummits, deltaCwithsummits)

write.table(MTBP_summits, "bed_files/MTBP_summits.bed", quote = F,
            col.names = F, row.names = F, sep = "\t")
```

Look at the intersection of WT and deltaC. Obtain output of bed files. These were used to create Figure 2C.

```
# Make bed files that have WT_only, WT_deltaC, and deltaC_only peaks.
bedtools intersect -a WT.bed -b deltaC.bed -u -wa > WT_deltaC.bed;
bedtools intersect -a WT.bed -b deltaC.bed -v > WT_only.bed;
bedtools intersect -a deltaC.bed -b WT.bed -v > deltaC_only.bed;
bedtools intersect -a WT_summits.bed -b WT_only.bed -wa > WT_only_summits.bed;
bedtools intersect -a WT_summits.bed -b WT_deltaC.bed -wa \
> WT_deltaC_summits.bed;
bedtools intersect -a deltaC_summits.bed -b deltaC_only.bed -wa \
> deltaC_only_summits.bed;
# WT and deltaC
computeMatrix reference-point --referencePoint center -R WT_deltaC_summits.bed \
WT_only_summits.bed deltaC_only_summits.bed \
-S WT.bw deltaC.bw --missingDataAsZero -a 3000 -b 3000 --binSize 10 \
-out Matrix_WT_deltaC.tab.gz

plotHeatmap -m Matrix_WT_deltaC.tab.gz -out hm_WT_deltaC.pdf \
--colorMap YlGnBu --refPointLabel "Peak"  # Figure 2C.
```

Make combined peak file “MTBP.bed”.

```
cat bed_files/WT.bed bed_files/intervene/WT_deltaC/sets/01_deltaC.bed > \
bed_files/MTBP.bed  ## All peaks.
```

Figure 2D

Compare MTBP-WT and MTBP-deltaC.

```
computeMatrix reference-point --referencePoint center -p max -R MTBP_summits.bed \
-S WT.bw deltaC.bw --missingDataAsZero -a 1000 -b 1000 --binSize 10 \
-out Figure2D.tab.gz;

plotProfile -m Figure2D.tab.gz -out Figure2D.pdf  --samplesLabel WT deltaC \
--refPointLabel "MTBP Peak Summits" --colors darkblue darkblue;
```

Peak Annotation

Annotate peaks was performed using gencode_basic annotation file obtained from https://www.gencodegene. “MTBP_peaks_1.txt” file was created.

```
annotatePeaks.pl bed_files/MTBP.bed hg38 -gtf gencode.v32.basic.annotation.gtf \
-go ../GO  > MTBP_peaks_1.txt
## Annotate peaks output was also used for Figure S2D.
```

Figure S2C

Peaks were quantified using Deeptools multiBigwigSummary.

```
multiBigwigSummary BED-file -b WT.bw cont.bw -out results_WT.npz --BED MTBP.bed \
--outRawCounts WT_counts.tab

multiBigwigSummary BED-file -b deltaC.bw cont.bw -out results_deltaC.npz \
--BED MTBP.bed -p max --outRawCounts deltaC_counts.tab

# Subtract control counts from the samples. Skip header.
tail WT_counts.tab -n+1 | awk 'OFS="\t"{print $1, $2, $3, $4-$5}' > WT-cont.tab
tail deltaC_counts.tab -n+1 | awk 'OFS="\t"{print $1, $2, $3, $4-$5}' > \
deltaC-cont.tab

# Combine the files with MTBP.bed file.
bedtools intersect -a MTBP.bed -b WT-cont.tab -wa -wb > MTBP_count_WT.bed
bedtools intersect -a MTBP.bed -b deltaC-cont.tab -wa -wb > MTBP_count_deltaC.bed
```

In R, calculate the Peak Scores. The output from multiBigwigSummary is average score in the peak. The bin size = 10.

```
library(ggpubr)
library(ggpmisc)

WT_counts <- read.delim("bed_files/MTBP_count_WT.bed", header = F) %>%
    select(c(1:10, 14)) %>% rename(PeakID = V4, Peak.score = V5) %>%
    mutate(WT.score = V14*(V3-V2)/10) %>% select(c(1:5), WT.score)

deltaC_counts <- read.delim("bed_files/MTBP_count_deltaC.bed", header = F) %>%
    select(c(1:10, 14)) %>% rename(PeakID = V4, Peak.score = V5) %>%
    mutate(deltaC.score = V14*(V3-V2)/10) %>% select(PeakID, deltaC.score)

MTBP_counts <- WT_counts %>% left_join(deltaC_counts, by = "PeakID")
MTBP_counts$deltaC.score[MTBP_counts$deltaC.score < 0] <- 0
# Remove negative values.
MTBP_counts$WT.score[MTBP_counts$WT.score < 0] <- 0

# Regression model with intercept at 0.
WT_deltaC_lm <- lm(deltaC.score ~  0 + WT.score, data = MTBP_counts)
summary(WT_deltaC_lm)

# Sample 5000 data points for the figure.
subset <- sample_n(MTBP_counts, 5000)
FigS2C <- ggscatter(subset, size = 0.5, x = "WT.score", y = "deltaC.score",
                    color = "dodgerblue", alpha = 0.4, xlim = c(0, 1500),
                    ylim= c(0, 750)) +
  geom_abline(slope = WT_deltaC_lm$coefficients, size = 0.2)
```

Figure 2E

The difference in read counts per peak between MTBP-WT and MTBP-deltaC.

```
library(tidyverse)
library(ggpubr)

MTBP_counts <- read.delim("../MTBP_peaks.txt") %>% select(1, 9, 10) %>% gather(cell_l
```

```
ines, score, 2:3, factor_key = TRUE)

Fig2E <- ggviolin(MTBP_counts, x = "cell_lines", y = "score", ylim = c(0, 1500), draw
_quantiles = 0.5, fill = "cell_lines", alpha = 0.5, size =0.5,
               palette =c("#9BCD9B", "#FFC125"), xlab = "MTBP", width= 0.9, legend =
"") +
   stat_summary(fun = "mean", geom="text", aes(label=..y..), position = position_nudg
e(x=0.5, y=100)) +
    stat_compare_means(label.y =1000, label.x = 0.55, label = "p.format", method = "w
ilcox", method.args = list(altenative = "greater"), paired = TRUE)
```

MTBP_peak.txt file

"MTBP_peak.txt" file with summit information and peak scores was created.

```
MTBP1 <- read.delim("MTBP_peaks_1.txt", header = T) %>% select(c(1:4,6,9,10,16))

MTBP1$PeakID <- MTBP1[,1]

# Combine the WT and deltaC peak scores and the locations of the summits.
MTBP1 <- MTBP1 %>% select(PeakID, c(2:9))

MTBP_summits <- read.delim("bed_files/summits/MTBP_summits.bed", header = F)  %>%
    rename(PeakID = V4,summit.start = V2,summit.end = V3) %>% select(-V1)

MTBP2 <- MTBP1 %>%
    left_join(MTBP_counts, by = "PeakID") %>%
    select(c(1:8, 13, 14)) %>%
    left_join(MTBP_summits, by = "PeakID")

write.table(MTBP2, "MTBP_peaks.txt", sep = "\t", quote = F, col.names = TRUE,
            row.names = F )
```

Processing of Various Files

Processing of Files Used in Figures 3 and 4 Is Described.

Enhancer/Super-enhancer

Enhancer/Super-enhancer file is from Hnisz et al. (2013). https://www.sciencedirect.com/science/article/pii/S0092867413012270?via%3Dihub#mmc2 The file is hg19. The bed files were lifted over to hg38. The centers of enhancers were determined ("enhancer_center_sorted.bed").

```
bedtools intersect -a MTBP.bed -b refs/Enhancer_HCT116/hg38_Enhancer.bed -wa -u > \
MTBP_enhancer.bed

bedtools intersect -a MTBP.bed -b refs/Enhancer_HCT116/hg38_SuperEnhancer.bed -wa \
-u > MTBP_superenhancer.bed

awk 'OFS="\t"{print $4, "Enhancer"}' MTBP_enhancer.bed > MTBP_enhancer.txt

awk 'OFS="\t"{print $4, "SuperEnhancer"}' MTBP_superenhancer.bed > \
MTBP_superenhancer.txt

cat MTBP_enhancer.txt MTBP_superenhancer.txt > \
MTBP_SuperEnhancerEnhancer.txt

# Distance to Enhancers from the MTBP peak summits was calculated.
awk 'OFS="\t"{print $1,int($2+($3-$2)/2), int($2+($3-$2)/2)+1}' hg38_Enhancer.bed \
|bedtools sort > enhancer_center.bed # Center of the enhancers.

bedtools closest -a MTBP_summits_sorted.bed -b enhancer_center.bed -D a > \
MTBPtoEnhancer.bed
# Distance from the MTBP peaks to the center of the enhancers.
```

TSS

TSS information was obtained from gencode.v32.basic.annotation.gtf file from Gencode. https://www.gencodegenes.org/human/

```
# Obtain rows containing genes and remove chrM.  Converted from GTF (1-based) to bed
files (0-based).
awk 'OFS="\t"{if ($3 == "gene"){print $1, $4-1, $5, $14, $6, $7 }}' \
gencode.v32.basic.annotation.gtf | awk 'OFS="\t" {if ($1!="chrM"){print$0}}' > \
gencode.genes.bed  # 60572 genes.

# Genes.gtf to TSS.
awk 'OFS="\t" {if($6 == "-") {$2 = $3 - 1} else {$3 = $2 + 1} print}' \
gencode.genes.bed |awk -F'"' '$1=$1' OFS="\t" | \
awk 'OFS="\t" {print $1, $2, $3, $4,$6,$7}' >TSS1.bed
```

```
# Make bed file covering 1 kb upstream and 1 kb downstream of TSS.
awk 'OFS="\t" {print $1, $2-1000, $3+1000, $4,$5,$6}' TSS.bed > TSS_2000.bed
# 1kb upstream and 1kb downstream.
```

G4

G4 structures were predicted with G4Hunter (https://doi.org/10.1093/nar/gkw006) using hg38 and threshold = 2. The results were exported as a bed file ("G4H_hg38_2_ref.bed").

```
# TSS with or without G4. Figure 3C was made using TSS_G4H.bed and TSS_noG4H.bed.
bedtools intersect -a TSS_2000.bed -b G4H_hg38_2_ref.bed -wa -u > TSS_2000_G4H.bed
bedtools intersect -a TSS.bed -b TSS_2000_G4H.bed -wa > TSS_G4H.bed
bedtools intersect -a TSS.bed -b TSS_G4H.bed -wa -v > TSS_noG4H.bed

# MTBP with G4.
bedtools intersect -a bed_files/MTBP.bed -b G4H_hg38_2_ref.bed -wa -u > MTBP_G4H.bed
```

Transcription

RNA-seq file (total RNA from DLD-1) was downloaded from GEO (#GSE85688).
Rokavec M et al. Cancer Res 2017;77(8):1854-1867. PMID: 28130225

```
library(dplyr)
tss <- read.delim("TSS.bed", header = F) %>% rename(Name =V4)
RNAseq <- read.delim("GSE85685_Rokavec_processed_data.txt",
                     header = T) %>%
    select(Name, DLD1) # Column "DLD1" contains the RNA_seq value (RPKM).

tss_RNA <- tss %>% left_join(RNAseq, by = "Name")
# TSS file and RNA-seq file were joined and essential elements were selected.

tss_RNA$rank <- NA
order.rank <- order(tss_RNA$DLD1, decreasing = T)
tss_RNA$rank[order.rank] <- 1:nrow(tss_RNA)
# Each entry was numbered according to expression level.

tss_RNA_select <- tss_RNA[complete.cases(tss_RNA),] %>% arrange(rank)
#  Rows containing NA were removed.

write.table(tss_RNA_select, "TSS_RNA.bed", quote = F, row.names = F, col.names = F,
            sep = "\t")
# All genes with RNA info were arranged from high to no expression.
```

AP-1 site

AP-1 sites were mapped using HOMER scanMotifGenomeWide.pl. Other motif files (AP1ver2(TTASTCA), RUNX1(AACCACA), TEAD(GGAATGY)) were made in the same manner.

```
# Create motif file with zero mismatch.
seq2profile.pl TGAGTCA 0 AP1 > TGAGTCA.motif

# Make bed file containing motifs.
scanMotifGenomeWide.pl TGAGTCA.motif hg38 -bed -mask | \
awk 'OFS = "\t"{print $1, $2-1, $3,$4,$5,$6}' > AP1.bed

# Find MTBP peaks with AP-1 motif.
bedtools intersect -a bed_files/MTBP.bed -b AP1.bed -wa -u > MTBP_AP1.bed

# Make bigwig file.
sort -k1,1 -k2,2n AP1.bed | bedItemOverlapCount hg38 -chromSize=hg38.chrom.sizes \
stdin > AP1.bedgraph;
bedGraphToBigWig AP1.bedgraph hg38.chrom.sizes AP1.bw
```

Combine the location information into MTBP_peaks.txt files in R.

```
library(dplyr)
MTBP2 <- read.delim("MTBP_peaks.txt", header = TRUE)
Enh <- read.delim("bed_files/MTBP_SuperEnhancerEnhancer.txt",
                  header = F) %>% rename(PeakID = V1, Enhancer = V2)

MTBP3 <- MTBP2 %>% left_join(Enh, by="PeakID")  %>%
    distinct(PeakID, .keep_all = TRUE)

# Change NA to "no".
MTBP3$Enhancer <- as.character(MTBP3$Enhancer)
MTBP3$Enhancer[is.na(MTBP3$Enhancer)] <- "no"

# Add Location information to the column.
MTBP4 <- MTBP3 %>%
    mutate(TSS = case_when(Distance.to.TSS < 2000 & Distance.to.TSS > -2000 ~ "TSS",
                           TRUE ~ "no"))  %>%
    mutate(Location = case_when(TSS == "TSS" ~ "Promoter_TSS",
                                TSS != "TSS" & Enhancer != "no" ~
                                    "Enhancer_SuperEnhancer",
                                TSS != "TSS" & Enhancer == "no" ~ "Others")) %>%
    select(-TSS, -Enhancer)

write.table(MTBP4, "MTBP_peaks_2.txt", sep = "\t", quote = F,
            col.names = TRUE, row.names = F )
```

Figure 3

Figure 3A

```
# Pie plot for location of MTBP peaks.
library(ggplot2)
library(dplyr)
library(ggsci)
library(scales)
MTBP <- read.delim("MTBP_peaks.txt", header = TRUE)
MTBP %>% group_by(Location) %>% tally()  # The values of this output were used in the
next line.

dat <- data.frame(MTBP = c("Promoter-TSS", "Enhancer Super-enhancer", "Others"),
                  value = c(10365, 5257, 14236)) %>%
    mutate(prop = value/(10365+5257+14236)*100) %>%
    mutate(lab.ypos = cumsum(prop) - 0.5*prop)

dat$prop <- round(dat$prop, digits = 1)

dat$MTBP <- factor(dat$MTBP, levels = c("Others", "Enhancer Super-enhancer",
                                        "Promoter-TSS"))

Fig3A <-ggplot(dat, aes(x="", y = prop, fill = MTBP)) +
    geom_bar(width = 1, stat = "identity", color = "white") +
    coord_polar("y", start = 0) + theme_void() +
    geom_text(aes(y = lab.ypos, label = percent(prop/100)), size = 5,
              color = "white")  +
    scale_fill_aaas(breaks = c("Promoter-TSS", "Enhancer Super-enhancer",
                               "Others"))
```

Figures 3B, 3C, 3D, and S3B

Figures 3B, 3C, and 3D were made using Deeptools computeMatrix and plotHeatmap. Figure S3B was made using Deeptools computeMatrix and plotProfile.

```
computeMatrix reference-point --referencePoint center \
-R Promoter_TSS_summits.bed Enhancer_Superenhancer_summits.bed Other_summits.bed \
-S WT.bw deltaC.bw Orc2.bw BG4.bw  --missingDataAsZero -a 3000 -b 3000 --binSize 10 \
-out Matrix_Figure3B.tab.gz

plotHeatmap -m Matrix_Figure3B.tab.gz -out Figure3B.pdf --colorMap YlGnBu YlGnBu \
Blues Blues --refPointLabel "Peak" --sortUsingSamples 4 --whatToShow "heatmap only"
```

Figures 3C and 3D were made using “TSS_G4H.bed”, “TSS_noG4H.bed”, and “TSS_RNA.bed” files. Figure S3B was made using “TSS.bed”.

Figure S3C

Correlation between transcription and MTBP peaks at TSS.

```r
library(dplyr)
library(ggpubr)
MTBP4 <- read.delim("MTBP_peaks.txt", header = T)
TSS_RNA <- read.delim("TSS_RNA.bed", header = F, sep ="\t") %>%
    select(c(1:4, 7 ,8)) %>%
    rename(Gene.Name = V4, Transcription = V7, rank = V8)

MTBP_RNA <- MTBP4 %>% left_join(TSS_RNA, by = "Gene.Name") %>%
    filter(Location == "Promoter_TSS")  # Filter the MTBP peaks at Promoter_TSS.

data <- MTBP_RNA[complete.cases(MTBP_RNA),]  #  Selected rows with RNA data.

FigS3C <-ggscatter(data, x = "Transcription", y = "WT.score", color = "Blue",
                   size = 0.5, alpha = 0.4, ylim = c(80, 1500)) +
    xscale("log10") + yscale("log10")
```

Figure 4

Figure 4A

```
library(dplyr)
library(ggpubr)
library(ggsci)
library(cowplot)

MTBP <- read.delim("../MTBP_peaks.txt", header = TRUE)

# Add distance to Enhancer data.
DistancetoEnhancer <- read.delim("MTBPtoEnhancer.bed", header = F) %>%
    select(V4, V8) %>% rename(PeakID = V4, Distance.to.Enhancer = V8) %>%
    distinct(PeakID, .keep_all = TRUE)
MTBP4 <- MTBP %>% left_join(DistancetoEnhancer, by = "PeakID")

# Add information on G4.
G4H <- read.delim("MTBP_G4H.bed", header = FALSE) %>% select(PeakID = V4)
G4H$G4 <- "G4"
MTBP5 <- MTBP4 %>% left_join(G4H, by = "PeakID")

MTBP5$G4[is.na(MTBP5$G4)] <- "no"

# Add information on AP-1 to the MTBP file.
AP1 <- read.delim("MTBP_AP1.bed", header = F) %>% select(PeakID = V4)
AP1$AP1 <- "TGAGTCA"

MTBP6 <- MTBP5 %>% left_join(AP1, by = "PeakID")
MTBP6$AP1[is.na(MTBP6$AP1)] <- "no"

df_TSS_10000 <- MTBP6 %>% filter(Distance.to.TSS > -10000 & Distance.to.TSS < 10000)

df_Enhancer_20000 <- MTBP6 %>%
    filter(Distance.to.Enhancer > -20000 & Distance.to.Enhancer < 20000)

a <-ggdensity(df_TSS_10000, x = "Distance.to.TSS", y = "..density..", fill = "G4",
              color ="G4", alpha = 0.5, xlim = c(-10000, 10000),
              palette = "startrek", xlab = "Distance to TSS (bp)")
a2 <- ggpar(a, font.x = c(12, "bold"), font.y = c(14, "bold"),
            font.xtickslab = c(12, "bold"),  font.ytickslab = c(12, "bold"),
            ylab = "Density", font.legend = c(14, "bold"))
b <- ggdensity(df_Enhancer_20000, x = "Distance.to.Enhancer", y = "..density..",
               fill = "G4", color ="G4", alpha = 0.5, palette = "startrek",
               xlim = c(-20000, 20000), xlab = "Distance to Enhancer (bp)")
b2 <- ggpar(b, font.x = c(12, "bold"), font.y = c(14, "bold"),
            font.xtickslab = c(12, "bold"),  font.ytickslab = c(12, "bold"),
            ylab = "Density",font.legend = c(14, "bold"))
# Figure 4A top
ggarrange(a2, b2, common.legend = TRUE, legend = "right")

c <- ggdensity(df_TSS_10000, x = "Distance.to.TSS", y = "..density..",
               fill = "AP1", color ="AP1", alpha = 0.5, palette = "jco",
```

```
                xlim = c(-10000, 10000), xlab = "Distance to TSS (bp)")
d <- ggdensity(df_Enhancer_20000, x = "Distance.to.Enhancer", y = "..density..",
                fill = "AP1", color ="AP1", alpha = 0.5, palette = "jco",
                xlim = c(-20000, 20000), xlab = "Distance to Enhancer (bp)")
c2 <- ggpar(c, font.x = c(12, "bold"), font.y = c(14, "bold"),
            font.xtickslab = c(12, "bold"),  font.ytickslab = c(12, "bold"),
            ylab = "Density", font.legend = c(14, "bold"))
d2 <- ggpar(d, font.x = c(12, "bold"), font.y = c(14, "bold"),
            font.xtickslab = c(12, "bold"),  font.ytickslab = c(12, "bold"),
            ylab = "Density", font.legend = c(14, "bold"))
# Figure 4A bottom
ggarrange(c2, d2, common.legend = TRUE, legend = "right")

# Count the numbers of the samples in each group.
df_TSS_10000 %>% group_by(G4) %>% tally()
df_TSS_10000 %>% group_by(AP1) %>% tally()
df_Enhancer_20000 %>% group_by(G4) %>% tally()
df_Enhancer_20000 %>% group_by(AP1) %>% tally()
```

Perform Kolmogorov-Smirnov test (Supplementary Table S1)

```
library(dplyr)
TSS_G4 <- df_TSS_10000 %>% filter(G4 == "G4")
TSS_noG4 <- df_TSS_10000 %>% filter(G4 != "G4")

Enh_G4 <- df_Enhancer_20000 %>% filter(G4 == "G4")
Enh_noG4 <- df_Enhancer_20000 %>% filter(G4 != "G4")

TSS_AP1 <- df_TSS_10000 %>% filter(AP1 == "TGAGTCA")
TSS_noAP1 <- df_TSS_10000 %>% filter(AP1 != "TGAGTCA")

Enh_AP1 <- df_Enhancer_20000 %>% filter(AP1 == "TGAGTCA")
Enh_noAP1 <- df_Enhancer_20000 %>% filter(AP1 != "TGAGTCA")

TSS.G4 <- ks.test(TSS_G4$Distance.to.TSS, TSS_noG4$Distance.to.TSS)
Enh.G4 <- ks.test(Enh_G4$Distance.to.Enhancer, Enh_noG4$Distance.to.Enhancer)

TSS.AP1 <- ks.test(TSS_AP1$Distance.to.TSS, TSS_noAP1$Distance.to.TSS)
Enh.AP1 <- ks.test(Enh_AP1$Distance.to.Enhancer, Enh_noAP1$Distance.to.Enhancer)
```

Figure 4B

```
# Obtain MTBP summits in Promoter-TSS, Enhancer/Super-enhancer, and Others.
library(readr)
summits <- MTBP6 %>% select(Chr, summit.start, summit.end, Location)
summits %>% group_by(Location) %>%
    do(write_tsv(., paste0(unique(.$Location), "_summits.bed"), col_names = FALSE))
# This creates three bed files with summits of MTBP peaks.

computeMatrix reference-point --referencePoint center \
-R Promoter_TSS_summits.bed Enhancer_SuperEnhancer_summits.bed Others_summits.bed \
-S AP1.bw --missingDataAsZero -a 1000 -b 1000 --binSize 10 -out Figure4B.tab.gz
```

```
plotProfile -m Figure4B.tab.gz -out Figure4B.pdf --refPointLabel "MTBP" \
--numPlotsPerRow 1 --perGroup --colors darkblue darkblue darkblue
```

Figure 4C

```
library(tidyr)
t <- MTBP6 %>% group_by(Location, AP1, G4) %>% tally()

all <- MTBP6 %>% group_by(AP1, G4) %>% tally()
all$Location <- "All"

bound <- bind_rows(all, t)

all_data <- all %>% bind_rows(t) %>%
  mutate(Motifs = case_when(AP1=="no"&G4 =="no" ~ "None",
                            AP1!="no" & G4!="no" ~ "Both",
                            AP1!="no"&G4=="no" ~ "AP1",
                            G4!="no"&AP1=="no"~"G4")) %>% rename(Counts = "n")

all_data$Motifs <- factor(all_data$Motifs,levels = c("G4", "Both", "AP1", "None"))

all_data$Location <- factor(all_data$Location,
                            levels = c("All", "Promoter_TSS",
                                       "Enhancer_SuperEnhancer", "Others"))

Fig4C <- ggbarplot(all_data, x="Location", y="Counts",  size = 0, ylab = FALSE,
               xlab=FALSE, order = c("Others", "Enhancer_SuperEnhancer",
                                     "Promoter_TSS", "All"), alpha = 0.8,
               fill = "Motifs", palette = c("#0060FF", "#00b0e0", "#00FFC0",
                                            "#C7c7c7"), color = "white",
               orientation = "horiz")
```

Figure S4C

```
# Obtain summits in MTBP peaks containing AP-1 motifs.
# Also obtain matrix of AP-1 motifs near MTBP peaks.
bedtools intersect -a MTBP_summits.bed -b MTBP_AP1.bed -wa > MTBP_summits_AP1.bed

computeMatrix reference-point --referencePoint center -R MTBP_summits_AP1.bed \
-S AP1.bw --missingDataAsZero -a 1000 -b 1000 --binSize 10 \
-out Matrix_MTBP_AP1.tab.gz
gunzip Matrix_MTBP_AP1.tab.gz
```

Look at matrix in R.

```
library(ggpmisc)
library(tidyr)
library(cowplot)
```

```r
# Read the matrix file.
TGAGTCA_matrix <- read.delim("../Matrix_MTBP_AP1.tab", skip = 3, header = F)  %>%
    select(-V4, -V5, -V6) %>%
    rename(Chr = V1, Start = V2, End = V3)

# Separate rows with the AP-1 motif on the right, left, or both sides.
TGAGTCA <- TGAGTCA_matrix %>%
    mutate(left=select(., V7:V106) %>% apply(1, sum,na.rm=TRUE)) %>%
    mutate(right=select(., V107:V206) %>% apply(1, sum,na.rm=TRUE)) %>%
    mutate(AP1_site = case_when(left == right ~ "both", left>right ~ "left",
                                right>left ~ "right")) %>%
  filter(right != 0 | left !=0)

a <- seq(from = -995, to = 995, by = 10) # Make x-axis
TGAGTCA_all <- colSums(TGAGTCA[,c(4:203)])/nrow(TGAGTCA)

df <- data.frame(data.frame(a, TGAGTCA_all))  # Combine x-axis and values.

# Plots
FigS4C1 <- ggplot(df, aes(x = a, y = TGAGTCA_all)) +
    geom_smooth(span = 0.02, se = F, color = "darkblue") +
    stat_peaks(geom = "text", hjust =1.5, color = "red", ignore_threshold = 0.5) +
    xlab("Distance from MTBP Summits (bp)") + ylab("AP-1 Motif") +
    theme_cowplot() + panel_border(size = 1, linetype = 1)

TGAGTCA_group <- TGAGTCA %>% group_by(AP1_site) %>%
    summarize_at(.vars= c(4:203), list(sum))
# Summarize the 4th to 203rd columns to obtain sum.

TGAGTCA %>% group_by(AP1_site) %>% tally()
# 862 for both, 3693 for left, and 3663 for right peaks.  10.5 % have both. 89.5% for
either side.

TGAGTCA %>% select(c(1:3, 206)) %>% group_by(AP1_site) %>%
  group_walk(~ write.table(.x, paste0(.y$AP1_site, "_MTBP_TGAGTCA.bed"),
                           sep="\t", quote=F, col.names = F, row.names=F))
## right_MTBP_TGAGTCA.bed and left_MTBP_TGAGTCA.bed files were made.

group <- data.frame(t(TGAGTCA_group[, -1]))
colnames(group) <- c("both", "left", "right")

group <- group %>% bind_cols(df) %>% select(1:4) %>% rename(distance = a)

group_long <- group %>% gather(key = "location", value = "count", -distance) %>%
    mutate(count = count/8218)
# Transform the data frame to long format. 8218 is the number of rows.

FigS4C2 <- ggplot(group_long, aes(x = distance, y = count, group = location,
                                  color = location)) +
    geom_smooth(span = 0.02, se = F) + theme_cowplot() +
    panel_border( size = 1, linetype = 1) + theme(legend.position=c(0.8, 0.8)) +
    xlab("Distance from MTBP Summits (bp)") + ylab("AP-1 Motif") +
    scale_color_manual(values = c("gray", "red", "blue"))

left <- group_long %>% filter(location == "left")
```

```
right <- group_long %>% filter(location  == "right")

FigS4C3 <- ggplot(left, aes(x = distance, y = count, group = location,
                            color = location)) +
    geom_smooth(span = 0.02, se = F, color = "red") +
    theme_cowplot() + panel_border( size = 1, linetype = 1) + ylim(0, 0.020) +
    xlab("Distance from MTBP Summits (bp)") + ylab("AP-1 Motif")

FigS4C4 <- ggplot(right, aes(x = distance, y = count, group = location,
                             color = location)) +
    geom_smooth(span = 0.02, se = F, color = "blue") +
    theme_cowplot() + panel_border( size = 1, linetype = 1) +
    ylim(0, 0.020) +
    xlab("Distance from MTBP Summits (bp)") + ylab("AP-1 Motif")
# bin size = 10  145 bp from the center.
```

Figure S4D

Look at peaks at TSS to orient peaks.

```
#  Make TSS.bw file.
sort -k1,1 -k2,2n TSS.bed | bedItemOverlapCount hg38 -chromSize=hg38.chrom.sizes \
stdin > TSS.bedgraph

bedGraphToBigWig TSS.bedgraph hg38.chrom.sizes TSS.bw
# Look at TSS placement around Promoter_TSS_summits files.

computeMatrix reference-point --referencePoint center -R Promoter_TSS_summits.bed \
-S TSS.bw --missingDataAsZero -a 1000 -b 1000 --binSize 10 \
-out Matrix_MTBP_TSS.tab.gz
gunzip Matrix_MTBP_TSS.tab.gz
```

Read the matrix in R.

```
TSS_matrix <- read.delim("../Matrix_MTBP_TSS.tab", skip = 3, header = F)  %>%
    select(-V4, -V5, -V6) %>% rename(Chr = V1, Start = V2, End = V3)

TSS <- TSS_matrix %>%
    mutate(left=select(., V7:V106) %>% apply(1, sum,na.rm=TRUE)) %>%
    mutate(right=select(., V107:V206) %>% apply(1, sum,na.rm=TRUE)) %>%
    mutate(TSS = case_when(left == right ~ "both", left>right ~ "left",
                           right>left ~ "right")) %>%
  filter(left != 0 | right !=0)

a <- seq(from = -995, to = 995, by = 10) # Make x-axis
TSS_all <- colSums(TSS[,c(4:203)])/nrow(TSS)

df <- data.frame(data.frame(a, TSS_all))  # Combine x-axis and values.

FigS4D1 <- ggplot(df, aes(x = a, y = TSS_all)) +
    geom_smooth(span = 0.05, se = F, color = "darkblue") +
    stat_peaks(geom = "text", hjust =1.5, color = "red", ignore_threshold = 0.98) +
    xlab("Distance from MTBP Summits (bp)") + ylab("TSS") +
```

```
    theme_cowplot() + panel_border(color = "black", size = 1, linetype = 1)

# Do exactly the same for TSS as for AP-1 motif.
TSS_group <- TSS %>% group_by(TSS) %>% summarize_at(.vars= c(4:203), list(sum))
# Summarize the 4th to 203rd columns to get sum.

TSS %>% group_by(TSS) %>% tally()  # 805 for both, 3331 for left, and 3463 for right
peaks.  10.6 % have TSS on both sides. 89.4% on either side.
TSS %>% select(c(1:3, 206)) %>% group_by(TSS) %>%
    group_walk(~ write.table(.x, paste0(.y$TSS, "_MTBP_TSS.bed"), sep="\t",
                             quote=F, col.names = F, row.names=F))

group_TSS <- data.frame(t(TSS_group[, -1]))
colnames(group_TSS) <- c("both", "left", "right")

group_TSS_2 <- group_TSS %>% bind_cols(df) %>% select(1:4) %>% rename(distance = a)

group_long_TSS <- group_TSS_2 %>% gather(key = "location", value = "count",
                                         -distance) %>%
    mutate(count = count/8428) # Transform the data frame to long format.

FigS4D2 <- ggplot(group_long_TSS, aes(x = distance, y = count, group = location,
                                      color = location)) +
    geom_smooth(span = 0.05, se = F) + theme_cowplot() +
    panel_border(color = "black", size = 1, linetype = 1) +
    theme(legend.position=c(0.8, 0.8)) + xlab("Distance from MTBP Summits (bp)") +
    ylab("TSS") + scale_color_manual(values = c("gray", "red", "blue"))

left_TSS <- group_long_TSS %>% filter(location == "left")
right_TSS <- group_long_TSS %>% filter(location  == "right")
both_TSS <- group_long_TSS %>% filter(location == "both")

FigS4D3 <- ggplot(left_TSS, aes(x = distance, y = count, group = location,
                                color = location)) +
    geom_smooth(span = 0.05, se = F, color = "red") + theme_cowplot() +
    panel_border(color = "black", size = 1, linetype = 1) + ylim(0, 0.002) +
    xlab("Distance from MTBP Summits (bp)") + ylab("TSS")

FigS4D4 <- ggplot(right_TSS, aes(x = distance, y = count, group = location,
                                 color = location)) +
    geom_smooth(span = 0.05, se = F, color = "blue") + theme_cowplot() +
    panel_border(color = "black", size = 1, linetype = 1) + ylim(0, 0.002) +
    xlab("Distance from MTBP Summits (bp)") + ylab("TSS")
# bin size = 10  145 bp from the center.
```

Figure 5

Figure 5C

```
cat ../AP1.bed ../G4H_hg38_2_ref.bed > AP1_G4.bed
bedtools slop -i ../bed_files/H3K4me2_DLD1.bed -b 200 -g ../../../hdd/refs/hg38.chrom
.sizes > H3K4me2_400bp_DLD1.bed

bedtools intersect -a H3K4me2_400bp_DLD1.bed -b AP1_G4.bed -wa -u > H3K4me2_AP1_G4.be
d # 33549 peaks for H3K4me2 with G4 or AP-1.
bedtools intersect -a H3K4me2_400bp_DLD1.bed -b AP1_G4.bed -wa -v > H3K4me2_noAP1_noG
4.bed # 28955 peaks for H3K4me2 without G4 or AP-1.
bedtools intersect -a H3K4me2_400bp_DLD1.bed -b ../AP1.bed -wa -u > H3K4me2_AP1.bed;
bedtools intersect -a H3K4me2_400bp_DLD1.bed -b ../G4H_hg38_2_ref.bed -wa -u > H3K4me
2_G4.bed;
bedtools intersect -a H3K4me2_400bp_DLD1.bed -b ../bed_files/summits/MTBP_summits.bed
-wa -u > H3K4me2_MTBP_summits.bed
```

Put data into R and analyze.

```
library(tidyverse)
library(ggpubr)

H3K4me2_400 <- read.delim("H3K4me2_400bp_DLD1.bed", header = F) %>% rename(ID = V4)
H3K4me2_AP1 <- read.delim("H3K4me2_AP1.bed", header = F)%>% rename(ID = V4) %>% selec
t(ID)
H3K4me2_AP1$AP1 <- "AP1"

H3K4me2_G4 <- read.delim("H3K4me2_G4.bed", header = F)%>% rename(ID = V4) %>% select(
ID)
H3K4me2_G4$G4 <- "G4"

H3K4me2_MTBP_summit <- read.delim("H3K4me2_MTBP_summits.bed", header = F)%>% rename(I
D = V4) %>% select(ID)
H3K4me2_MTBP_summit$MTBP <- "MTBP"

H3K4me2 <- H3K4me2_400 %>%  left_join(H3K4me2_AP1, by = "ID") %>%  left_join(H3K4me2_
G4, by = "ID")%>%  left_join(H3K4me2_MTBP_summit, by = "ID")
H3K4me2[is.na(H3K4me2)] <- "no"

H3K4me2_motifs <- H3K4me2[, c(1:4, 11:13)] %>% mutate(any_motifs = case_when(AP1=="no
" & G4=="no" ~ "without",
                                                                              AP1=="AP
1" | G4=="G4" ~ "with"))
# Compare H3K4me2 with or without the motifs for co-localization with MTBP peaks.
H3K4me2_a <- H3K4me2_motifs %>%  group_by(any_motifs, MTBP) %>% tally() %>% rename(Co
unts = "n")
H3K4me2_a$any_motifs <- factor(H3K4me2_a$any_motifs, levels = c("with", "without"))
H3K4me2_a$MTBP <- factor(H3K4me2_a$MTBP, levels = c("MTBP", "no"))


Fig5C <- ggbarplot(H3K4me2_a, x="any_motifs", y="Counts",  size = 0, ylab = FALSE, xl
ab = FALSE,
```

```
           order = c("with", "without"), alpha = 0.8,
          fill = "MTBP", palette = c("#3333FF", "#009999")) + scale_y_continuous(expa
nd = c(0,0))

mat1<- H3K4me2_a %>% spread(MTBP, Counts)
mat2 <- mat1[,-1]
row.names(mat2)<- c("with_motif", "without_motif")
# Fisher's exact test
test <- fisher.test(mat2)
test
```

Figure 5D

```
computeMatrix reference-point --referencePoint center -p max -R Promoter_TSS_summits.
bed  Enhancer_Superenhancer_summits.bed Other_summits.bed -S  ../bw_files/WT.bw ../bw
_files/deltaC.bw  ../bw_files/H3K4me2_DLD1.bw  --missingDataAsZero -a 1000 -b 1000 --
binSize 10 \
-out Matrix_WT_deltaC_H3K4me2_2.tab.gz

plotHeatmap -m Matrix_WT_deltaC_H3K4me2_2.tab.gz -out Fig5D.pdf --colorMap  YlGnBu Yl
GnBu Blues Greens --refPointLabel "Peak" --sortUsingSamples 1 --zMax 20 20 80  --what
ToShow "heatmap and colorbar"
```

Figures 5E and 5F

Use left_MTBP_TGAGTCA.bed and right_MTBP_TGAGTCA.bed to make a new bed file with strand orientations in column 6. Make right peaks as "+" and left as "-".

```
awk 'OFS = "\t"{print $0,".",".", "+"}' right_MTBP_TGAGTCA.bed > right_MTBP.bed
awk 'OFS = "\t"{print $0,".",".","-"}' left_MTBP_TGAGTCA.bed > left_MTBP.bed

cat right_MTBP.bed left_MTBP.bed | sort -k 1,1 -k2,2n > MTBP_TGAGTCA_dir.bed #7356

awk 'OFS = "\t"{print $0,".",".", "+"}' right_MTBP_TSS.bed > right_TSS_MTBP.bed;
awk 'OFS = "\t"{print $0,".",".","-"}' left_MTBP_TSS.bed > left_TSS_MTBP.bed;

cat right_TSS_MTBP.bed left_TSS_MTBP.bed | sort -k 1,1 -k2,2n > MTBP_TSS_dir.bed
```

Figures S5C and S5D

Use these oriented files (MTBP_TGAGTCA_dir.bed and MTBP_TSS_dir.bed) to make matrix files for Figures S5C and S5D (FigureS5D.tab and FigureS5D.tab)

```
# Figure S5C
computeMatrix reference-point --referencePoint center -R MTBP_TGAGTCA_dir.bed \
-S WT.bw MNase_hg38.bw H3K4me2_DLD1.bw H3K4me1.bigWig H3K4me2.bigWig H3K4me3.bigWig H
3K9Ac.bigWig \
H3K27ac.bw DNase.bw YY1.bigWig --missingDataAsZero -a 1000 -b 1000 --binSize 10 \
-out direcMTBP.tab.gz
```

```
# Figure S5C matrix
plotProfile -m direcMTBP.tab.gz -o dir_MTBP_AP1.pdf --refPointLabel \
"MTBP Peak Summit" --yMin 0 0.06 3 3 3 2 2 0 0     --yMax 12 0.1 40 6 10 8 6 8 1.8 6
\
--regionsLabel " "  --samplesLabel MTBP MNase-seq H3K4me2_DLD1 H3K4me1 H3K4me2 H3K4me
3 H3K9ac \
H3K27ac DNase-seq YY1 --outFileNameData FigureS5C.tab

# Figure S5D
computeMatrix reference-point --referencePoint center -R MTBP_TSS_dir.bed \
-S WT.bw MNase_hg38.bw H3K4me2_DLD1.bw H3K4me1.bigWig H3K4me2.bigWig H3K4me3.bigWig H
3K9Ac.bigWig \
H3K27ac.bw DNase.bw YY1.bigWig G4H.bw --missingDataAsZero -a 1000 -b 1000 \
--binSize 10 -out dir_MTBP_TSS.tab.gz

# Figure S5D matrix
plotProfile -m dir_MTBP_TSS.tab.gz -o dir_MTBP_TSS.pdf --yMax 12 0.14 40 4 20 50 20 1
5 4 15 0.05 --yMin 0 0.05 0 1 9 10 2 5 0 0 0.005  --refPointLabel "MTBP Peak Summit"
\ --regionsLabel " " --samplesLabel MTBP MNase-seq H3K4me2_DLD H3K4me1 H3K4me2 H3K4me
3 H3K9ac \
H3K27ac DNase-seq YY1 G4 --outFileNameData FigureS5D.tab;
```

Figures 5E and 5F

```
library(tidyr)
library(tibble)
library(dplyr)
library(ggpubr)
library(viridis)
df <- read.delim("FigureS5C.tab", skip = 1, header = F)
data <- df[,c(1,3:202)]

norm_data <- t(apply(data[2:10,-1], 1, function(x)(x-min(x))/(max(x)-min(x))))
# The data was normalized from 0 to 1.

# Transform the matrix to the long format.
a <- norm_data %>% as_tibble() %>%
  rowid_to_column(var="X") %>%
  gather(key="Y", value="Z", -1) %>%  mutate(Y=as.numeric(gsub("V","",Y)))

Fig5E <- ggplot(a, aes(X, Y, fill= Z)) +
    geom_tile() +
    theme(legend.position="none") +
    scale_fill_viridis(discrete=FALSE, option = "D", begin = 0, end =0.7) +
    theme_bw()+ coord_flip()+ scale_x_reverse()

# Do the same analysis with TSS.
df_TSS <- read.delim("FigureS5D.tab", skip = 1, header = F)
data_TSS <- df_TSS[,c(3:202)]
rownames(data_TSS) <- df_TSS[,1]
```

```r
# Smooth G4H data.
data_TSS_t <- as.data.frame(t(data_TSS))  # Rows and columns are transposed.

G4H_smooth <- loess(G4 ~ bins, data = data_TSS_t, span = 0.1)
G4H_smoothed <- predict(G4H_smooth)
# Check the smoothing.
ggscatter(data_TSS_t, x = "bins", y = "G4") + geom_smooth(span = 0.1)

# Add smoothed G4 data to the data frame (data_TSS).
newdata <- rbind(data_TSS[1:11,], G4H_smoothed)
rownames(newdata)[12] <- "G4H_smoothed"

# Normalize and draw the graph.
norm_data_TSS <- t(apply(newdata[2:12,], 1, function(x)(x-min(x))/(max(x)-min(x))))
# The data was normalized from 0 to 1.
a_TSS <- norm_data_TSS %>% as_tibble() %>%
  rowid_to_column(var="X") %>%
  gather(key="Y", value="Z", -1) %>%  mutate(Y=as.numeric(gsub("V","",Y)))

Fig5F <- ggplot(a_TSS, aes(X, Y, fill= Z)) +
    geom_tile() +
    theme(legend.position="none") +
    scale_fill_viridis(discrete=FALSE, option = "D", begin = 0, end =0.7) +
    theme_bw()+ coord_flip()+ scale_x_reverse()
```

Figure 6

Processing of the files from EdU-seq experiments

Making EdU_seq.bw files (Figure 6A and Figure S6B)

The three replicates of files were aligned to the human genome in the same manner as CUT&RUN samples.

```
# Made bw files using Deeptools bamCoverage as follows:
for file in *.bam; do bamCoverage -b ${file} -o bw_files/${file/%.bam/.bw} -bs 50 \
--normalizeUsing RPGC -p max --ignoreDuplicates --effectiveGenomeSize 2913022398 \
--blackListFileName hg38_blacklist.bed
# This makes three bw files (rep1, rep2, rep3) for each of EdU_seq_WT_rep.bw, EdU_seq
_deltaC_rep.bw, and EdU_cont_rep.bw.

# Comparison between WT and control bigwig files and deltaC and control bigwig files.
# Control is EdU-seq experiments using G1-arrested cells.
# The values are log2 ratio (sample/control).  bin = 2000
bigwigCompare -b1 EdU_seq_WT_rep1.bw -b2 EdU_cont_rep1.bw -o WT_cont_rep1.bw \
-bs 2000 --blackListFileName hg38_blacklist.bed
# Do this for each replicate. (Also, for each of EdU_seq_deltaC_rep.bw files.)

# Comparison between replicates.
# Figure S6B
multiBigwigSummary bins -b WT_cont_rep1.bw WT_cont_rep2.bw WT_cont_rep3.bw \
-o multibigwig_WT.npz;

plotCorrelation -in multibigwig_WT.npz --corMethod pearson --whatToPlot scatterplot \
-o FigS6A.pdf --labels rep1 rep2 rep3;

# Merge the files.  The scores are average of three files.
bigWigMerge WT_cont_rep1.bw WT_cont_rep2_WT.bw WT_cont_rep3.bw EdU-seq.bedgraph;

awk 'OFS = "\t"{print $1, $2, $3, $4/3}' EdU-seq.bedgraph | sort -k 1,1 -k2,2n \
> EdU-seq_sorted.bedgraph;
# Score is log2ratio of WT over control averaged between three replicates.

bedGraphToBigWig EdU-seq_sorted.bedgraph hg38.chrom.sizes  EdU_seq_WT.bw
# Do the above for EdU_seq_deltaC.bw
```

Use HOMER to call peaks.

```
makeTagDirectory EdU_seq_WT_rep1/ -sspe EdU_seq_WT_rep1.bam;
makeTagDirectory EdU_cont_rep1/ -sspe EdU_cont_rep1.bam;
# Repeat for other two replicates.

findPeaks EdU_seq_WT_rep1/ -i EdU_cont_rep1 -region -size 6000 -minDist 12000 -F 2 >\
EdU1.txt
# Repeat this for other two replicates.
```

```
# Merge peaks of three replicates.
mergePeaks -d given EdU1.txt EdU2.txt EdU3.txt -prefix EdU

grep -v '^#' EdU_EdU1.txt_EdU2.txt EdU_EdU1.txt_EdU3.txt EdU_EdU2.txt_EdU3.txt \
EdU_EdU1.txt_EdU2.txt_EdU3.txt > Initiation_zones.txt

pos2bed.pl Initiation_zones.txt > Initiation_zones.bed # Initiation zones.
```

Figure S6C

Check the size of the initiation zones.

```
library(dplyr)
library(ggpubr)

# Initiation zones = Origins
Origins <- read.delim("Initiation_zones.bed", header = F, skip = 1) %>% mutate(length
= V3-V2) %>%
    rename(OriginID = V4, Chr = V1) %>% mutate(size=length/1000)
# 2473 very early initiation zones.

mean(Origins$length)  # 57.8 kb (OK-seq : mean size = 30 kb, 6-150 kb)
median(Origins$length) # 48 kb

FigS6C <- gghistogram(Origins$size, add = "median",
                      add.params = list(color = "red"), xlim = c(4, 250),
                      bins = 50, xlab = "Size (kb)", color = "dodgerblue2",
                      fill = "dodgerblue1",
                      font.label = list(size = 14, face = "bold"))

## Calculate the length of the origins.
total_length <- sum(Origins$size)  # kb
# 143116282  Effective genome size hg38 2913022398
143116282/2913022398 * 100  # 4.9% of the genome.
```

Figure 6C

```
# Compare the EdU-seq-WT, EdU-seq-deltaC, and EdU-control read counts.
multiBigwigSummary BED-file -b EdU_seq_WT_rep1.bw EdU_seq_WT_rep2.bw \ EdU_seq_WT_rep
3.bw EdU_seq_deltaC_rep1.bw EdU_seq_deltaC_rep2.bw \ EdU_seq_deltaC_rep3.bw EdU_cont_
rep1.bw EdU_cont_rep2.bw EdU_cont_rep3.bw \ --outRawCounts WT_deltaC_EdU.tab -p max -
-BED Initiation_zones.bed -o EdU_WT_deltaC.npz
# This yields a file (WT_deltaC_EdU.tab) containing log2 average reads for each of
the initiation zones.
```

Put the data in R.

```
library(tidyverse)
library(ggpubr)
library(ggpmisc)

EdU_WT_deltaC <- read.delim("WT_deltaC_EdU.tab", header = TRUE )

EdU_data <- setNames(EdU_WT_deltaC, c("chr", "start", "end", "WT1", "WT2", "WT3",
                                      "deltaC1","deltaC2", "deltaC3", "cont1",
                                      "cont2", "cont3")) %>%
    mutate(size = end-start) %>%
    mutate(WT = (WT1+WT2+WT3)/3, deltaC = (deltaC1+deltaC2+deltaC3)/3,
           cont = (cont1+cont2+cont3)/3) %>%
    mutate(WT_cont = WT - cont, deltaC_cont = deltaC-cont) %>%
    select(chr, start, end, size, WT_cont, deltaC_cont) %>%
    mutate(ratio = deltaC_cont/WT_cont) %>% mutate(ID = row_number())

mean(EdU_data$WT_cont)
mean(EdU_data$deltaC_cont)

EdU <-gather(EdU_data, group, average_reads, WT_cont:deltaC_cont, factor_key = TRUE)

Fig6C <- ggboxplot(EdU, x = "group", y = "average_reads", ylab = "Average Reads",
                   fill = "#CD534CFF", outlier.shape = NA, ylim = c(0,7)) +
    scale_y_continuous(expand = c(0, 0)) +
    stat_compare_means(paired = TRUE, label.y = 5, label.x = 2, label ="p.format")
```

Figure 6D

Make control bed file (10,000 data points) with regioneR.

```
library(regioneR)
control_summits<- createRandomRegions(nregions=10000, length.mean = 2,
                                      length.sd = 0, genome= "hg38", mask=NULL)
write.table(toDataframe(control_summits), file = "control_summits.bed",
            sep="\t", col.names = FALSE, row.names = FALSE, quote=FALSE)

# Initiation zones around MTBP
computeMatrix reference-point --referencePoint center -R Promoter_TSS_summits.bed \
Enhancer_SuperEnhancer_summits.bed Others_summits.bed control_summits.bed  \
-S EdU_seq.bw --missingDataAsZero -a 100000 -b 100000 --binSize 1000 \
-out RO_MTBP2.tab.gz;

plotProfile -m RO_MTBP2.tab.gz -out Fig6D.pdf --regionsLabel "Promoter_TSS" \
"Enhancer_SuperEnhancer" "Others" "random" --samplesLabel EdU-seq --yMin 0 \
--yMax 0.6 --colors green red darkblue yellow --legendLocation upper-left \
--refPointLabel "Center"
```

Figures 6E and 6F

Look at MTBP peaks around TSS and group them according to the location of summits of MTBP relative to TSS. (5′side of TSS or 3′ side of TSS) TSS_2000.bed file is 1 kb upstream and 1 kb downstream of TSS. MTBP summits that are within this region are selected.

```
bedtools intersect -a TSS_2000.bed -b MTBP_summits.bed -wa -wb > TSS_MTBP_peaks.bed
bedtools intersect -a TSS_2000.bed -b MTBP_summits.bed -wa -v > TSS_noMTBP.bed
```

Separate the MTBP peaks with summits on the 5′ side of TSS and 3′ side of TSS.

```
library(dplyr)
library(readr)
TSS_MTBP <- read.delim("../TSS_MTBP_peaks.bed", header = F)

TSS_MTBP_plus <- TSS_MTBP %>% filter(V6 == "+") %>%
    mutate(loc = case_when(V8 > V2 + 1000 ~ "right", TRUE ~ "left"))
# V8 is the summit of MTBP. "V2 + 1000" is TSS.

TSS_MTBP_minus <- TSS_MTBP %>% filter(V6 == "-") %>%
    mutate(loc = case_when(V8 > V3 - 1000 ~ "left", TRUE ~ "right"))

TSS_MTBP_loc <- bind_rows(TSS_MTBP_plus, TSS_MTBP_minus) %>%
    mutate(start = as.integer(V2 + 1000), end = as.integer(start + 1)) %>%
    select(V1, start, end, V4, V5, V6, loc)

TSS_MTBP_loc_right <- TSS_MTBP_loc %>% filter(loc == "right")
TSS_MTBP_loc_left <- TSS_MTBP_loc %>% filter(loc == "left")

write.table(TSS_MTBP_loc_right, "TSS_MTBP_loc_right.bed", sep = "\t",
            col.names = F, row.names = F, quote = F)
# TSS with MTBP summits on the 3' end.

write.table(TSS_MTBP_loc_left, "TSS_MTBP_loc_left.bed", sep = "\t",
            col.names = F, row.names = F, quote = F)
# TSS with MTBP summits on the 5' end.

tss <- read.delim("../TSS.bed", header = F)
```

Draw the graph of MTBP peaks (6E) and EdU-seq reads (6F).

```
# MTBP signal around TSS (Figure 6E)
computeMatrix reference-point --referencePoint TSS -R TSS_MTBP_loc_right.bed \
TSS_MTBP_loc_left.bed -S /WT.bw --missingDataAsZero -a 1000 -b 1000 --binSize 10 \
-out Matrix_TSS_MTBP_rl.tab.gz;

plotProfile -m Matrix_TSS_MTBP_rl.tab.gz -out Figure6D.pdf --regionsLabel "3'" "5'" \
--samplesLabel MTBP --colors darkblue cyan

# Initiation zones around TSS (Figure 6F)
computeMatrix reference-point --referencePoint TSS -p max -R TSS_MTBP_loc_right.bed \
TSS_MTBP_loc_left.bed TSS_noMTBP.bed -S EdU_seq.bw --missingDataAsZero -a 50000 \
-b 50000 --binSize 200 -out TSS_RO_MTBP_rl.tab.gz;

plotProfile -m TSS_RO_MTBP_rl.tab.gz -out Figure6F.pdf \
```

```
--regionsLabel "3'" "5'" "no MTBP" --samplesLabel EdU-seq \
--outFileNameData Figure6E.tab
```

Calculate the EdU peak summits from TSS. Sharp peaks around TSS (<500 bp) and broad peaks around 10 kb are observed.

```
library(dplyr)
library(ggpubr)
library(ggpmisc)
library(tidyr)
library(cowplot)
edu_peaks <- read.delim("../Figure6E.tab", header = F, skip = 1)
edu_df <- t(edu_peaks)
edu <- data.frame(edu_df[c(-1, -2),], stringsAsFactors = FALSE)
edu$X1 <- as.integer(edu_df[3:502,1])
edu <- edu %>% mutate(bins = X1-250)
colnames(edu) <- c("X1", "right", "left", "noMTBP", "bins")
edu$right <- as.numeric(edu$right)
edu$left <-as.numeric(edu$left)
edu$noMTBP <- as.numeric(edu$noMTBP)

ggplot(edu, aes(x = bins, y = right)) +
  geom_smooth(span = 0.01, se = F, color = "darkblue") +
  stat_peaks(geom = "text", hjust =1.5, color = "red", ignore_threshold = 0.708) +
  stat_valleys(geom = "text", hjust =1, color = "green", ignore_threshold = 0.3) +
  xlab("Distance from TSS (bins)") +
  theme_cowplot() + panel_border(color = "black", size = 1, linetype = 1)

ggplot(edu, aes(x = bins, y = left)) +
    geom_smooth(span = 0.01, se = F, color = "darkblue") +
    stat_peaks(geom = "text", hjust =1, color = "red", ignore_threshold = 0.81) +
    stat_valleys(geom = "text", hjust =1, color = "green", ignore_threshold = 0.1) +
    xlab("Distance from TSS (bins)") + theme_cowplot() +
    panel_border(color = "black", size = 1, linetype = 1)
# bin = 200
```

Replication Timing Data

The files for HCT-116 were downloaded from https://www2.replicationdomain.com/database.php. The two files were merged.

```
bedtools unionbedg -filler "NA" -i RT_HCT116_1.bdg RT_HCT116_2.bdg > merged_RT.txt
# The first few lines were edited out.
awk 'OFS="\t"{print $1, $2, $3, ($4+$5)/2}' merged_RT.txt > RT_HCT116.bdg
```

Segmentation was performed using DNAcopy in Bioconductor.

```
library(DNAcopy)

RT <- read.delim("RT_HCT116.bdg", header = F, sep = "\t")
RT$LOC <- as.numeric(RT$V2)
repTiming <- CNA(RT$V4, RT$V1, RT$LOC, data.type ="logratio", sampleid = "RT")
```

```
Seg.RT <- segment(repTiming, nperm=10000, alpha=0.01,
                  undo.splits = "sdundo", undo.SD=65, verbose=2)
par(ask=T,mar=c(3.1,4.1,1,1))
plot(Seg.RT, plot.type="c")
plot(Seg.RT, plot.type="s")
plot(subset(Seg.RT,chromlist="chr2"), pch=19, pt.cols=c("gray","gray"),
     xmaploc=T,ylim=c(-4.5,4.5))

write.table(Seg.RT$output,"Seg_RT", row.names=F, quote=F, sep= "\t")

RT1  <- Seg.RT$output

# Fill the gaps between the segments.
RT <- RT1 %>% mutate(end = loc.end + 2499, start = loc.start - 2500) %>%
    mutate(size = end - start)
write.table(RT[,c("chrom", "start", "end", "num.mark", "seg.mean", "size")],
            "RT.bed", row.names=F, col.names=F, quote=F, sep="\t")
# RT.bed describes the chr, start, end, RT_scores, and the size of the segments.
```

Intersect RT.bed file with MTBP.bed file and count how many MTBP peaks are intersected with each replication timing segment.

```
bedClip RT.bed hg38.chrom.sizes RT_c.bed
# Domains are clipped to the size within chromosomes.

bedtools intersect -a RT_c.bed -b bed_files/MTBP.bed -c > RT_MTBP_count.bed
```

For each timing segment, if average RT_score was more than 1.75, the segment was designated as “Early”. If RT_score was less than -1.75, it was designated as “Late”. If the score was in between, it was designated as “Mid”.

```
library(ggsci)

RT_MTBP <- read.table("RT_MTBP_count.bed", header = F, sep = "\t")
colnames(RT_MTBP) <- c("chr", "start", "end", "data.count", "RT_score",
                       "length", "peak_count")

RT_MTBP$peaks_per_100kb <- (RT_MTBP$peak_count/RT_MTBP$length)*100000

RT_MTBP_T <- RT_MTBP %>%
    mutate(Timing = case_when(RT_score >= 1.75 ~ "Early",
                              RT_score < 1.75 & RT_score >= -1.75 ~ "Mid",
                              RT_score < -1.75 ~ "Late"))

length <- RT_MTBP_T %>% group_by(Timing) %>% tally(length, name = "length")

peak_count <- RT_MTBP_T %>% group_by(Timing) %>%
    tally(peak_count, name = "peak_count")

RT_MTBP_T$Timing <- factor(RT_MTBP_T$Timing, levels = c("Early", "Mid", "Late"))

# Add color to the segments for viewing in the browser.
RT_MTBP_C <- RT_MTBP_T %>%
    mutate(RGB = case_when(Timing =="Early" ~ "30,144,255",
                           Timing =="Mid" ~ "255,215,0",
```

```
                            Timing =="Late" ~ "169,169,169"))

RT_MTBP_C$strand <- "."

write.table(RT_MTBP_C[,c(1:5,11, 2,3,10)], "RT_color.bed", row.names=F,
            col.names=F, quote=F, sep="\t")
write.table(RT_MTBP_T[, c(1:3, 9)], "RT_summary.bed", row.names=F,
            col.names=F, quote=F, sep="\t")
```

Intersect with replication timing file and initiation zone.

```
bedtools intersect -a RT_summary.bed -b bed_files/Initiation_zones.bed  -c > \ RT_EdU
_count.bed
```

Figure S6A

```
# Add replication timing information to MTBP peak file.
bedtools intersect -a bed_files/MTBP.bed -b RT_summary.bed -wa -wb > MTBP_RT.bed
```

Add data in R and plot.

```
MTBP <- read.delim("MTBP_peaks.txt", header = TRUE)
MTBP_RT <- read.delim("MTBP_RT.bed", header = F) %>% select(PeakID = V4, RT = V14)
MTBP_timing <- MTBP %>% left_join(MTBP_RT, by = "PeakID") %>% select(Location, RT)
MTBP_timing <- MTBP_timing[complete.cases(MTBP_timing),]

MTBP_timing$Location <- factor(MTBP_timing$Location, levels = c("Promoter_TSS",
                                                                "Enhancer_SuperEnhancer",
                                                                "Others") )

MTBP_timing$RT <- factor(MTBP_timing$RT, levels = c("Early", "Mid", "Late") )
a <- MTBP_timing %>% group_by(RT, Location) %>% tally(n())

FigS6A <- ggbarplot(a, fill="Location", color = "white", y="n", x="RT",
                    position = position_dodge(0.8),
                    palette = c("#008B45","#EE0000","#3B4992"))
```

Figure 6B

```
Fig6B <- ggboxplot(RT_MTBP_T, x="Timing", y= "peaks_per_100kb",
                   xlab = "Replication Timing", ylab = "No. of Peaks/100kb",
                   fill = "Timing", ylim = c(0,5), palette = get_palette("jco", 3),
                   outlier.shape=NA) +
    stat_compare_means(method = "anova", label.x = "Mid", label.y =4)

# Calculate mean values.
RT_MTBP_T %>% group_by(Timing) %>% tally(mean(peaks_per_100kb))
```

Figure S6D

```
computeMatrix reference-point --referencePoint TSS -R TSS_MTBP_loc_right.bed \
TSS_MTBP_loc_left.bed -S bw_files/WT.bw bw_files/deltaC.bw Refs/MNase_hg38.bw H3K4me2
_DLD1.bw Refs/H3K4me1*.bigWig Refs/H3K4me2*.bigWig Refs/H3K4me3*.bigWig   Refs/H3K27a
c.bw Refs/H3K9Ac*.bigWig Refs/YY1*.bigWig \
--missingDataAsZero -a 1000 -b 1000 --binSize 10 \
-out Matrix_TSS_MTBP_rl_H3.tab.gz;

plotHeatmap -m Matrix_TSS_MTBP_rl_H3.tab.gz -out FigureS6D.pdf
```

Figure 7

Figure 7A

Make a control file for initiation zones using regioneR.

```
library(regioneR)

initiation_zone <- read.delim("bed_files/Initiation_zones.bed",
                              header = FALSE, skip=1) %>%
    select(c(1,2,3))

control_zone <- randomizeRegions(initiation_zone, genome= "hg38",
                                 allow.overlaps = FALSE)

write.table(toDataframe(control_zone), file = "control_initiation_zone.bed",
            sep="\t", col.names = FALSE, row.names = FALSE, quote=FALSE)
```

Count MTBP peaks and YY1 peaks in initiation zones.

```
# Count the number of MTBP peaks per initiation zone and control zone.
bedtools intersect -a bed_files/Initiation_zones.bed -b bed_files/MTBP.bed -c > Init_
zone_MTBP.bed
bedtools intersect -a control_initiation_zone.bed -b bed_files/MTBP.bed  \
-c > control_zone_MTBP.bed

# Count the number of YY1 peaks per initiation zone and control zone.
bedtools intersect -a bed_files/Initiation_zones.bed -b YY1.bed -c > Init_zone_YY1.be
d
bedtools intersect -a control_initiation_zone.bed -b YY1.bed \
-c > control_zone_YY1.bed
```

Draw graphs comparing YY1 and MTBP in initiation zones and control zones.

```
library(dplyr)
library(ggpubr)
library(ggsci)

Ini_MTBP <- read.delim("Init_zone_MTBP.bed", header = F) %>%
    rename(Chr = V1, Start= V2, End = V3, Count = V7) %>%
    select(Chr, Start, End, Count)

Ini_YY1 <- read.delim("Init_zone_YY1.bed", header = F) %>%
    rename(Chr = V1, Start= V2, End = V3, Count = V7) %>%
    select(Chr, Start, End, Count)

Cont_MTBP <- read.delim("control_zone_MTBP.bed", header = F) %>%
    rename(Chr = V1, Start= V2, End = V3, Count = V4)

Cont_YY1 <- read.delim("control_zone_YY1.bed", header = F) %>%
    rename(Chr = V1, Start= V2, End = V3, Count = V4)

MTBPcount <- bind_rows(Initiation_zone = Ini_MTBP,
                       Control = Cont_MTBP, .id = "groups") %>%
```

```
    mutate(length = End-Start) %>%
    mutate(peaks_per_100kb = (Count/length)*100000)

YY1count <- bind_rows(Initiation_zone = Ini_YY1,
                      Control = Cont_YY1, .id = "groups") %>%
    mutate(length = End-Start) %>%
    mutate(peaks_per_100kb = (Count/length)*100000)

Fig7Al <- ggboxplot(MTBPcount, x="groups", y= "peaks_per_100kb",
                    ylab = "No. of Peaks/100kb",  fill = "groups",
                    palette = c("#CD534CFF", "#7aa6DCFF"), outlier.shape=NA,
                    ylim = c(0, 10))  +
    stat_compare_means(label.y = 8, label.x = 2,label = "p.format")

Fig7Ar <- ggboxplot(YY1count, x="groups", y= "peaks_per_100kb",
                    ylab = "No. of Peaks/100kb",  fill = "groups",
                    palette = c("#CD534CFF", "#7aa6DCFF"), outlier.shape=NA,
                    ylim = c(0, 10))  +
    stat_compare_means(label.y = 8, label.x = 2,label = "p.format")

# Calculate the means.
MTBPcount %>% group_by(groups) %>% tally(mean(peaks_per_100kb))
YY1count %>% group_by(groups) %>% tally(mean(peaks_per_100kb))
```

Figures 7B and S7A

```
# Obtain the coordinates for the centers of initiation zones.
sorted_initiation_zone <- initiation_zone %>%
    mutate(chr = V1, size = V3-V2, center = V2 + as.integer(size/2)) %>%
    mutate(end = center + 1) %>%
    select(chr, center, end, size) %>%
    arrange(desc(size))
# Large to small initiation zones.

write.table(sorted_initiation_zone, "Initiation_center_sorted.bed", quote = F,
            sep = "\t", col.names=F, row.names =F )
# This file was used to create Heatmaps.
```

For heatmaps in Figure S7A, TSS file that covers 2000 bp downstream and upstream was used to increase the signal.

Figures 7C and S7B

YY1 HiChIP data is aligned to hg19 genome. MTBP.bw and EdU_seq data were converted to hg19. YY1_HiChIP data in HCT116 was downloaded from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM2774000.

```
library(regioneR)
library(dplyr)
```

```
library(readr)
YY1_HICHIP <- read.csv("GSM2774000_HCT116_YY1_hichip5kb-results.csv.gz",
                       header = TRUE)

# Select high confidence peaks and filter out interchromosomal interactions.
# Also, filter the PETs connecting adjacent bins.
yy1_loops <- YY1_HICHIP %>% filter(Bayes.mixture.1 > 0.9) %>%
    filter(chromosome1 == chromosome2) %>% filter(end1 != start2)

yy1_loops2 <- yy1_loops %>% mutate(LoopID = 1:n()) # Make LoopID
yy1_loops2$LoopID <- sprintf('Loop%i', yy1_loops2$LoopID)

yy1_loops3 <- yy1_loops2 %>% select(chromosome1, start1, end2, PET.Count, LoopID)
# Bed file with start of loop1 and end of loop2.

write.table(yy1_loops3, "yy1_start1_end2.bed", col.names = F, row.names = F,
            quote = F, sep = "\t")

# Make control regions of the same number and length using regioneR.
original <- read.delim("EdU_hg19.bed", header = F)
random <- randomizeRegions(original, genome= "hg19", allow.overlaps = FALSE)
write.table(toDataframe(random), file = "random.bed", sep="\t",
            col.names = FALSE, row.names = FALSE, quote=FALSE)
```

Intersect with initiation zones.

```
bedtools intersect -a EdU_hg19.bed -b yy1_start1_end2.bed -loj > overlapped_YY1.bed;
bedtools intersect -a random.bed -b yy1_start1_end2.bed \
-loj > overlapped_YY1_random.bed
```

Count the sum of PET scores per initiation zone.

```
detach(package:dplyr)
library(plyr)
library(dplyr)
library(ggpubr)
RO <- read.delim("../overlapped_YY1.bed", header = F)
random <- read.delim("../overlapped_YY1_random.bed", header = F)

# Count number of PETs.
RO_count <- RO %>% rename(Count = V7) %>%
    mutate(Count = case_when(V5 < V2-2500  | V6 > V3+2500  ~ "0",
                             Count == "." ~ "0",
                             TRUE ~ as.character(Count)))
# Remove PETs that stick outside of the initiation zone by more than 2500 bp.
# 2500 bp is half of the bin width (5000 bp).

random_count <- random %>% rename(Count = V7) %>%
    mutate(Count = case_when(V5 < V2-2500 | V6 > V3+2500  ~ "0",
                             Count == "." ~ "0",
                             TRUE ~ as.character(Count)))

RO_count$Count <- as.numeric(RO_count$Count)
random_count$Count <- as.numeric(random_count$Count)
```

```r
RO_data <- RO_count %>% group_by(V1, V2, V3) %>% summarize(n = sum(Count))

random_data <- random_count %>% group_by(V1, V2, V3) %>% summarize(n = sum(Count))

# Combine the two data frames.
df <- bind_rows(RO = RO_data, random = random_data, .id = "groups" ) %>%
    mutate(size = V3-V2) %>% mutate(PET_count_10K= (n/size)*10000) %>%
    filter(size >= 15000) # Small zones are removed.

df %>% group_by(groups) %>% summarise(mean(PET_count_10K))

Fig7C <- ggviolin(df, x = "groups", y = "PET_count_10K", outlier.shape = NA,
                  fill = "groups", palette = c("#CD534CFF", "#7aa6DCFF"), add = "non
e", draw_quantiles = 0.5)  +
    ylim(c(0, 20)) +
    stat_compare_means(label.y = 15, label.x = 1.5,label = "p.format")
```