Fig2. and Supplementary Fig2

#Fig.2 and Supp Fig.2

Analysis of RBP binding across all genes stratified by size of gene, and segmented into 1 kb bins.

```
library(dplyr)
library(tidyr)
library(ggplot2)
`%ni%` = Negate(`%in%`)
library(viridis)
library(forcats)
```

##wrangle data

group genes based on size

```
#wrangle data
readAndWrangleDataFrame <- function(FILEPATH){

  DF<-read.table(FILEPATH)
  colnames(DF)<-c("sample","chr","start","end","geneName","binNumber","strand","count")
  print(paste("read in ",FILEPATH))

  DF_1<-DF %>%
    select(-chr,-start,-end,-strand) %>%
    separate(sample, into=c("Protein","Rep","Timepoint")) %>%
    separate(geneName, into=c("geneName", "bioType", "exonID","genomicSize"), sep="\\:::") %>%
    mutate(genomicSize = as.numeric(genomicSize)) %>%
    mutate(sizeRange = case_when(
      genomicSize < 10000 ~ "<10k",
      genomicSize %in% c(10000:20000) ~ "10k-20k",
      genomicSize %in% c(20001:30000) ~ "20k-30k",
      genomicSize %in% c(30001:40000) ~ "30k-40k",
      genomicSize %in% c(40001:50000) ~ "40k-50k",
      genomicSize %in% c(50001:60000) ~ "50k-60k",
      genomicSize %in% c(60001:70000) ~ "60k-70k",
      genomicSize %in% c(70001:80000) ~ "70k-80k",
      genomicSize %in% c(80001:90000) ~ "80k-90k",
      genomicSize %in% c(90001:100000) ~ "90k-100k",
      genomicSize %in% c(100001:110000) ~ "100k-110k",
      genomicSize %in% c(110001:120000) ~ "110k-120k",
      genomicSize %in% c(120001:130000) ~ "120k-130k",
      genomicSize %in% c(130001:140000) ~ "130k-140k",
      genomicSize %in% c(140001:150000) ~ "140k-150k",
      genomicSize %in% c(150001:160000) ~ "150k-160k",
      genomicSize %in% c(160001:170000) ~ "160k-170k",
      genomicSize %in% c(170001:180000) ~ "170k-180k",
      genomicSize %in% c(180001:190000) ~ "180k-190k",
      genomicSize %in% c(190001:200000) ~ "190k-200k",
      genomicSize > 200000 ~ ">200k"
    ))

  return(DF_1)

}


#calculate coverage

calculateCoverage <- function(DF) {
 DF_1<- DF  %>%
    filter(!(Protein == "CBP20" & Rep == "3")) %>%
    left_join(mRNATotalBins) %>% #join total number of TUs for each bin
    #select the relevant columns (only one read type here)
    select(Protein, Rep, Timepoint, geneName,TotalBins, binNumber,sizeRange, count) %>%
    group_by(Protein, Rep, Timepoint,sizeRange) %>%
    mutate(n=n_distinct(geneName)) %>%
    ungroup() %>%
    group_by(Protein, Rep, Timepoint,binNumber, sizeRange,n) %>%
    summarise(binSum=sum(count),
              mean  =mean(count),
              mean_trim0.0025 = mean(count, trim = 0.0025)) %>%
    ungroup() %>%
    #factorise and add levels
    mutate( binNumber = as.numeric(as.character(binNumber)),
            Timepoint = factor(Timepoint, levels=c("negative", "PBSDRB", "t00","t05", "t10", "t15", "t2

  return(DF_1)
}
```

This part was run on the cluster for speed

```
# DF<-wrangled



# STEP_1 load and wrangle the data read and wrangle data
# from filepath input
# wrangled<-readAndWrangleDataFrame('tmp.intronic.counts')

# STEP_2 calculate total bins for each gene calculate total
# bins for each gene mRNATotalBins<-wrangled %>%
# select(geneName,binNumber, count) %>% group_by(geneName)
# %>% summarise(TotalBins = max(as.numeric(binNumber)))
# STEP_3 make a list of genes with over zero counts. this
# uses the total of all datasets.  make list of genes with
# over zero counts

# geneListOverZeroCounts<-wrangled %>% select(Protein,
# Timepoint, geneName, count) %>% group_by(geneName) %>%
# summarise(totalCounts = sum(as.numeric(count))) %>%
# filter(totalCounts > 0) %>% select(geneName) %>% unique()
# STEP_4 calculate coverage
# wrangled_calculated<-calculateCoverage(wrangled)
```

```
intron_st_maps <- read.table("../../data/all_introns.hg38_HeLa_Soren.1kbins.200409.processed_without_li
    header = T)
exon_and_intron_st_maps <- read.table("../../data/all_exon_and_intron.hg38_HeLa_Soren.1kbins.200409.pro
    header = T)

exon_and_intron_st_maps %>%
    select(Protein, Rep, Timepoint, n, sizeRange) %>%
    unique() %>%
    group_by(Protein, Rep, Timepoint) %>%
    summarise(n = sum(n))
```

```
## # A tibble: 96 x 4
## # Groups:   Protein, Rep [10]
##    Protein   Rep Timepoint     n
##    <fct>   <int> <fct>     <int>
##  1 ALYREF      1 DMSO      23152
##  2 ALYREF      1 negative  23152
##  3 ALYREF      1 PBSDRB    23152
##  4 ALYREF      1 t00       23152
##  5 ALYREF      1 t05       23152
##  6 ALYREF      1 t10       23152
##  7 ALYREF      1 t15       23152
##  8 ALYREF      1 t20       23152
##  9 ALYREF      1 t40       23152
## 10 ALYREF      1 t60       23152
## # ... with 86 more rows
```

```
sizeRanges_list = c("<10k", "10k-20k", "20k-30k", "30k-40k",
    "40k-50k", "50k-60k", "60k-70k", "70k-80k", "80k-90k", "90k-100k",
    "100k-110k", "110k-120k", "120k-130k", "130k-140k", "140k-150k",
    "150k-160k", "160k-170k", "170k-180k", "180k-190k", "190k-200k",
    "200k-210k", "210k-220k", "220k-230k", "230k-240k", "240k-250k",
    "250k-260k", "260k-270k", "270k-280k", "280k-290k", "290k-300k",
    ">300k")
unique(intron_st_maps$Timepoint)
```

```
##  [1] DMSO     negative PBSDRB   t00      t05      t10      t15      t20
##  [9] t40      t60
## Levels: DMSO negative PBSDRB t00 t05 t10 t15 t20 t40 t60
```

```
# add in levels

unique(intron_st_maps$sizeRange) %>%
    as.data.frame()
```

```
##              .
## 1         <10k
## 2        >300k
## 3    100k-110k
## 4      10k-20k
## 5    110k-120k
## 6    120k-130k
## 7    130k-140k
## 8    140k-150k
## 9    150k-160k
## 10 160k-170k
## 11 170k-180k
## 12 180k-190k
## 13 190k-200k
## 14 200k-210k
## 15   20k-30k
## 16 210k-220k
## 17 220k-230k
## 18 230k-240k
## 19 240k-250k
## 20 250k-260k
## 21 260k-270k
## 22 270k-280k
## 23 280k-290k
## 24 290k-300k
## 25   30k-40k
## 26   40k-50k
## 27   50k-60k
## 28   60k-70k
## 29   70k-80k
## 30   80k-90k
## 31  90k-100k
```

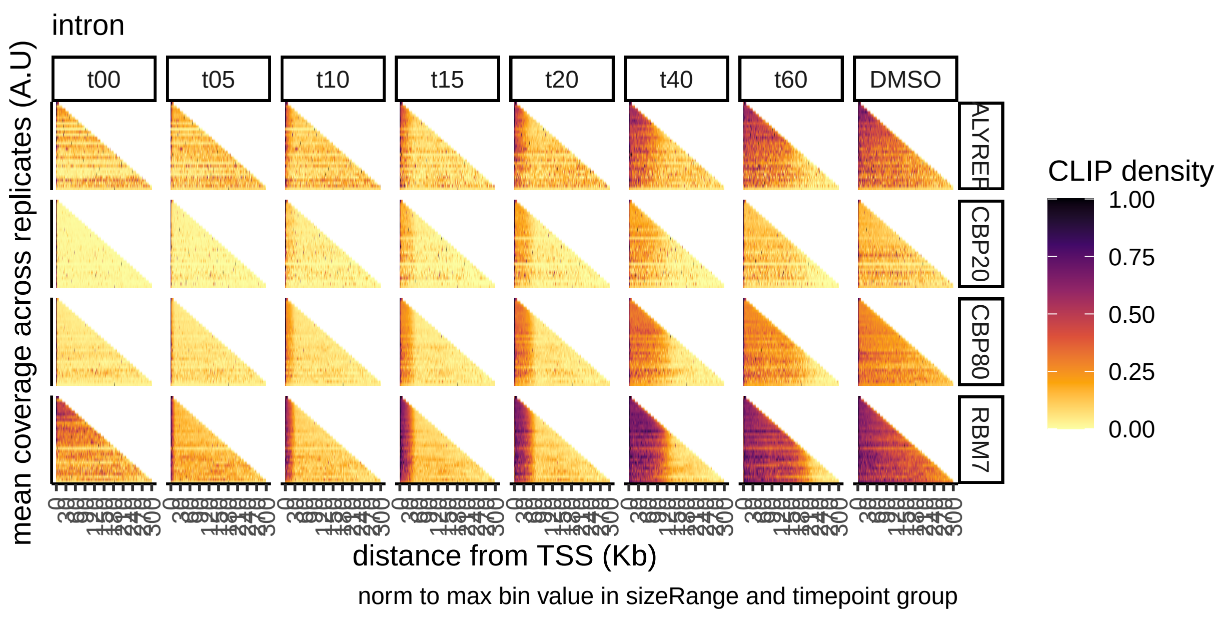
```
intron_st_maps <- intron_st_maps %>%
    mutate(genomicRegion = "intron")

exon_and_intron_st_maps <- exon_and_intron_st_maps %>%
    mutate(genomicRegion = "both")
# create long df with all data
all_st_maps <- rbind(intron_st_maps, exon_and_intron_st_maps) %>%
    mutate(Timepoint_f = case_when(Timepoint == "PBSDRB" ~ "t00",
        TRUE ~ as.character(Timepoint))) %>%
    mutate(sizeRange = factor(sizeRange, sizeRanges_list), Timepoint = factor(Timepoint,
        levels = c("negative", "PBSDRB", "t00", "t05", "t10",
            "t15", "t20", "t40", "t60", "DMSO")))

unique(all_st_maps$genomicRegion)
```

```
## [1] "intron" "both"
```

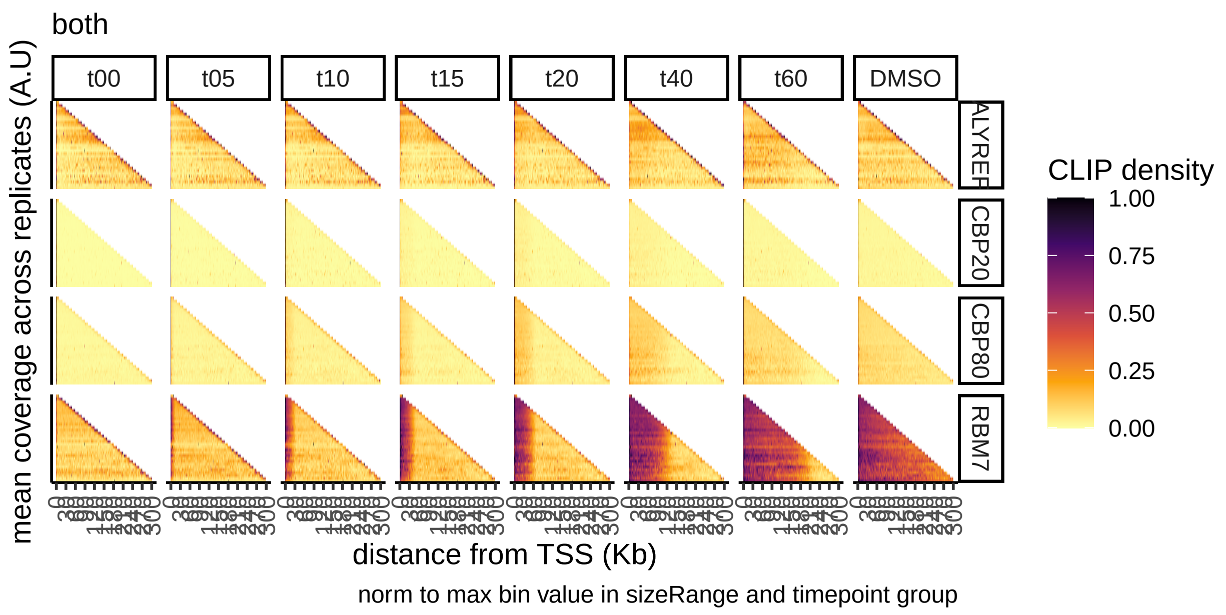
#Figure 2 A (only intron coverage) and Sup Fig 2 Cboth exon and intron)

```
region_list<-unique(all_st_maps$genomicRegion)

for(REGION in region_list){
  print(REGION)

  print(
    all_st_maps %>%
    filter(Timepoint_f %ni% c("negative") &
             genomicRegion == REGION ) %>%

    #first mean is to average the t00 and PBSDRB (these are the same timepoint)
    group_by(Protein, Timepoint_f, sizeRange, binNumber, genomicRegion, Rep) %>%
    summarise(mean_Rep = mean(mean)) %>%
    ungroup() %>%
    #second is to normalise the bin coverage by the max coverage amongt bins.
    #Basically : relativate to max coverage amongst that timepoint and sizeRange, so at least one bin w
    group_by(Protein, Timepoint_f, sizeRange, genomicRegion, Rep) %>%
    mutate(mean_rep_max = mean_Rep/max(mean_Rep)) %>%
    ungroup() %>%
    #final average is to take the average of the replicates.
    group_by(Protein, Timepoint_f, sizeRange, genomicRegion, binNumber) %>%
    summarise(mean_rep_max_rep = mean(mean_rep_max)) %>%
    ungroup() %>%
    mutate(Timepoint_f = factor(Timepoint_f, levels = c("t00", "t05","t10","t15","t20","t40","t60","DMS(
    ggplot() +
    geom_raster(aes(y=fct_reorder(sizeRange, desc(sizeRange)), x=as.numeric(binNumber), fill = as.numeri
    facet_grid(Protein~Timepoint_f) +
    labs(subtitle = paste0(REGION),
         caption = "norm to max bin value in sizeRange and timepoint group",
         fill = "CLIP density") +
    ylab("mean coverage across replicates (A.U)") +
    scale_x_continuous(name = "distance from TSS (Kb)",
                       breaks=seq(0,300, by=30),
                       limits = c(0,300),
    ) +
    scale_fill_viridis_c(option = "inferno", direction = -1, na.value="black") +
    theme_classic() +
    theme(
      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
      panel.background = element_blank(),
      axis.text.y=element_blank(),
      axis.ticks.y=element_blank(),
      panel.spacing = unit(0.25, "lines")
    )


  )
}
```

```
## [1] "intron"
```



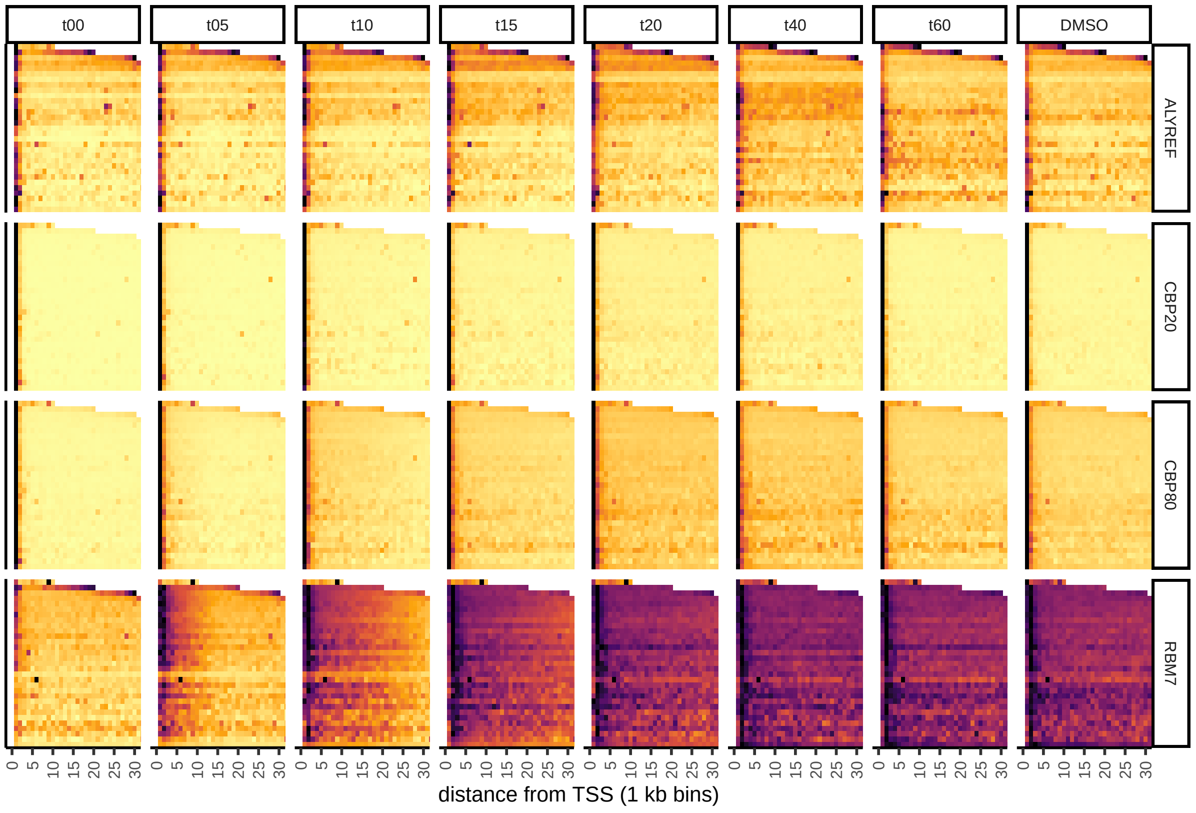
```
## [1] "both"
```



```r
#for zoom

region_list=list("both")
for(REGION in region_list){
  print(REGION)

  print(
 all_st_maps %>%
    filter(Timepoint_f %ni% c("negative") &
             genomicRegion == REGION ) %>%
    #first mean is to average the t00 adnd PBSDRB
    group_by(Protein, Timepoint_f, sizeRange, binNumber, genomicRegion, Rep) %>%
    summarise(mean_Rep = mean(mean)) %>%
    ungroup() %>%
    #second is to normalise the bin coverage by the max coverage amongt bins.
    #Basically : relativate to max coverage amongst that timepoint and sizeRange, so at least one bin w
    group_by(Protein, Timepoint_f, sizeRange, genomicRegion, Rep) %>%
    mutate(mean_rep_max = mean_Rep/max(mean_Rep)) %>%
    ungroup() %>%
    #final average is to take the average of the replicates.
    group_by(Protein, Timepoint_f, sizeRange, genomicRegion, binNumber) %>%
    summarise(mean_rep_max_rep = mean(mean_rep_max)) %>%
    ungroup() %>%
    #spread(binNumber, mean_rep_max_rep, fill = 0) %>%
    #gather("binNumber", "mean_rep_max_rep", c(`1`:`240`)) %>%
    mutate(Timepoint_f = factor(Timepoint_f, levels = c("t00", "t05","t10","t15","t20","t40","t60","DMS
    ggplot() +
    geom_raster(aes(y=fct_reorder(sizeRange, desc(sizeRange)), x=as.numeric(binNumber), fill = as.numer
    facet_grid(Protein~Timepoint_f) +
    labs(subtitle = paste0(REGION),
        # caption = "norm to max bin value in sizeRange and timepoint group\nfirst  10kb window",
         fill = "CLIP density") +
    ylab("") +
    scale_x_continuous(name = "distance from TSS (1 kb bins)",
                       breaks=seq(0,300, by=5),
                       limits = c(0,300),
    ) +
    scale_fill_viridis_c(option = "inferno", direction = -1, na.value="black") +
    theme_classic() +
    theme(text = element_text(size=8),
      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
      panel.background = element_blank(),
      axis.text.y=element_blank(),
      axis.ticks.y=element_blank(),
      panel.spacing = unit(0.25, "lines"),
      legend.position = "none"
    ) +
      coord_cartesian(xlim = c(0,30))
  )
}
```

```
## [1] "both"
```

both


t00
t05
t10
t15
t20
t40
t60
DMSO
ALYREF
CBP20
CBP80
RBM7
0
5
10
15
20
25
30
distance from TSS (1 kb bins)