

# Figure2 RAPseq

Qun Li

1 Sep 2024 ( 16:39:06 )

## Contents

0.1	ENV settings	1
0.2	R libraries	2
0.3	Colors settings	7
0.4	Motifs settings	8
0.4.1	eCLIP	8
0.4.2	RNA Compete	9
0.4.3	RBNS	11
0.4.4	RAPSeq	11
0.5	Figures settings	13
0.5.1	Figure 2A	13
0.5.2	Figure 2B	13
0.5.3	Figure 2C	16
0.5.4	Figure 2E	36
0.5.5	Figure 2F	38
0.5.6	Figure 2G	39
0.5.7	Figure S2B	52
0.5.8	Figure S2C	54
0.5.9	Figure S2D	56
0.5.10	Figure S2E	57
0.5.11	Figure S2F	60
0.5.12	Figure S2G	63

### 0.1 ENV settings

```

###  

# @Description: Figure2  

# @Description: Adapted from https://github.com/IonutAtanasoai1/RAPseq  

# @Author: LiQun  

# @Email: qun.li@ki.se  

# @Date: 1 Sep 2024 ( 16:39:06 )  

###  

rm(list = ls())  

setwd("/Users/liqun/Desktop/Projects/RAPseq/0_RAPSeq/")

```

## 0.2 R libraries

```

library(ChIPpeakAnno)  

## Loading required package: IRanges  

## Loading required package: BiocGenerics  

##  

## Attaching package: 'BiocGenerics'  

## The following objects are masked from 'package:stats':  

##  

##     IQR, mad, sd, var, xtabs  

## The following objects are masked from 'package:base':  

##  

##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,  

##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,  

##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,  

##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,  

##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,  

##     table, tapply, union, unique, unsplit, which.max, which.min  

## Loading required package: S4Vectors  

## Loading required package: stats4  

##  

## Attaching package: 'S4Vectors'  

## The following object is masked from 'package:utils':  

##  

##     findMatches  

## The following objects are masked from 'package:base':  

##  

##     expand.grid, I, unname  

## Loading required package: GenomicRanges  

## Loading required package: GenomeInfoDb  

## Warning: package 'GenomeInfoDb' was built under R version 4.3.3  

library(Biostrings)  

## Warning: package 'Biostrings' was built under R version 4.3.3  

## Loading required package: XVector  

##  

## Attaching package: 'Biostrings'  

## The following object is masked from 'package:base':  

##  

##     strsplit  

library(edgeR)

```

```

## Loading required package: limma
##
## Attaching package: 'limma'
## The following object is masked from 'package:BiocGenerics':
##
##      plotMA
library(ggfortify)
## Loading required package: ggplot2
library(tidyverse)
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyrr    1.3.1
## -- Conflicts ----- tidyverse_conflicts() --
## x lubridate::within%() masks IRanges::within%()
## x dplyr::collapse()   masks Biostrings::collapse(), IRanges::collapse()
## x dplyr::combine()   masks BiocGenerics::combine()
## x purrr::compact()   masks XVector::compact()
## x dplyr::desc()     masks IRanges::desc()
## x tidyrr::expand()   masks S4Vectors::expand()
## x dplyr::filter()   masks stats::filter()
## x dplyr::first()    masks S4Vectors::first()
## x dplyr::lag()      masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()   masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()   masks S4Vectors::rename()
## x lubridate::second() masks S4Vectors::second()
## x lubridate::second<-() masks S4Vectors::second<_()
## x dplyr::slice()    masks XVector::slice(), IRanges::slice()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(scales)
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##      discard
##
## The following object is masked from 'package:readr':
##
##      col_factor
library(ggplot2)
library(dendextend)
##
## -----
## Welcome to dendextend version 1.17.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues

```

```

## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
## To suppress this message use: suppressPackageStartupMessages(library(dendextend))
## -----
##
##
## Attaching package: 'dendextend'
##
## The following object is masked from 'package:Biostrings':
##
##      nnodes
##
## The following object is masked from 'package:stats':
##
##      cutree
library(dichromat)
library(reshape2)
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyverse':
##
##      smiths
library(dplyr)
library(stringr)
library(GenomicRanges)
library(GenomicFeatures)
## Warning: package 'GenomicFeatures' was built under R version 4.3.3
## Loading required package: AnnotationDbi
## Loading required package: Biobase
## Welcome to Bioconductor
##
##      Vignettes contain introductory material; view with
##      'browseVignettes()'. To cite Bioconductor, see
##      'citation("Biobase")', and for packages 'citation("pkgname")'.
##
##
## Attaching package: 'AnnotationDbi'
##
## The following object is masked from 'package:dplyr':
##
##      select
library(gplots)
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:IRanges':
##
##      space
##
## The following object is masked from 'package:S4Vectors':
##

```

```

##      space
##
## The following object is masked from 'package:stats':
##
##      lowess
library(clusterProfiler)
## Warning: package 'clusterProfiler' was built under R version 4.3.3
##
## clusterProfiler v4.10.1 For help: https://yulab-smu.top/biomedical-knowledge-mining-book/
##
## If you use clusterProfiler in published research, please cite:
## T Wu, E Hu, S Xu, M Chen, P Guo, Z Dai, T Feng, L Zhou, W Tang, L Zhan, X Fu, S Liu, X Bo, and G Yu.
##
## Attaching package: 'clusterProfiler'
##
## The following object is masked from 'package:AnnotationDbi':
##
##      select
##
## The following object is masked from 'package:purrr':
##
##      simplify
##
## The following object is masked from 'package:XVector':
##
##      slice
##
## The following object is masked from 'package:IRanges':
##
##      slice
##
## The following object is masked from 'package:S4Vectors':
##
##      rename
##
## The following object is masked from 'package:stats':
##
##      filter
library(org.Hs.eg.db)
##
library(org.Dr.eg.db)
##
library(tidyr)
library(gtools)
library(ggrepel)
library(ggbeeswarm)
library(ReactomePA)
## ReactomePA v1.46.0 For help: https://yulab-smu.top/biomedical-knowledge-mining-book/
##
## If you use ReactomePA in published research, please cite:
## Guangchuang Yu, Qing-Yu He. ReactomePA: an R/Bioconductor package for reactome pathway analysis and
library(VennDiagram)
## Loading required package: grid

```

```

## 
## Attaching package: 'grid'
##
## The following object is masked from 'package:Biostrings':
## 
##     pattern
## 
## Loading required package: futile.logger
##
## Attaching package: 'futile.logger'
##
## The following object is masked from 'package:gtools':
## 
##     scat
## 
## 
## Attaching package: 'VennDiagram'
##
## The following object is masked from 'package:dendextend':
## 
##     rotate
library(LSD)
library(gridExtra)
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:Biobase':
## 
##     combine
## 
## The following object is masked from 'package:dplyr':
## 
##     combine
## 
## The following object is masked from 'package:BiocGenerics':
## 
##     combine
library(UpSetR)
library(ggpubr)
##
## Attaching package: 'ggpubr'
##
## The following object is masked from 'package:VennDiagram':
## 
##     rotate
## 
## The following object is masked from 'package:dendextend':
## 
##     rotate
library(GenomicScores)
##
## Attaching package: 'GenomicScores'
##

```

```

## The following object is masked from 'package:utils':
##
##      citation
library(phastCons7way.UCSC.hg38)
## Warning: replacing previous import 'utils::findMatches' by
## 'S4Vectors::findMatches' when loading 'phastCons7way.UCSC.hg38'
library(phastCons100way.UCSC.hg38)
## Warning: replacing previous import 'utils::findMatches' by
## 'S4Vectors::findMatches' when loading 'phastCons100way.UCSC.hg38'
library(corrplot)
## Warning: package 'corrplot' was built under R version 4.3.3
## corrplot 0.94 loaded
library(ggforce)
library(idr)
library(GGally)
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg  ggplot2
library(eulerr)
## Registered S3 method overwritten by 'eulerr':
##   method from
##   plot.venn gplots
##
## Attaching package: 'eulerr'
##
## The following object is masked from 'package:gplots':
##
##      venn
library(ggseqlogo)
library(robustbase)
## Warning: package 'robustbase' was built under R version 4.3.3
##
## Attaching package: 'robustbase'
##
## The following object is masked from 'package:Biobase':
##
##      rowMedians
library(zoo)
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##      as.Date, as.Date.numeric

```

### 0.3 Colors settings

```

acqua_greens <- c("#04493F", "#02655B", "#08756A", "#0C8074", "#179286", "#25A296", "#3EB3A7", "#5FCBC1"
acqua_blues  <- c("#02465B", "#015666", "#0C6476", "#157589", "#1E8498", "#4397A8", "#4FB0C3", "#70C2D2"
greys <- c("#202020", "#404040", "#606060", "#808080", "#AOAOAO", "#COCOCO", "#EOEOEO", "#FFFFFF")
reds <- c("#C10303", "#D83535", "#E95F5F", "#F08686", "#FAAEAE")

```

```

pinks <- c("#660000", "#990000", "#CC0000", "#FF0000", "#FF6666", "#FF9999", "#FFCECE")
yellows <- c( "#9C8803", "#A59213", "#B09D1F", "#BAA82F", "#C6B441", "#D0BF53", "#DACP69", "#E4D782", "#F0E68C")
oranges <- c( "#A15000", "#CE7012", "#E47B12", "#F07D09", "#FF8B15", "#FFA141", "#FFCC99")

```

## 0.4 Motifs settings

```

## the following lines reconstructs the sequence motif logos from PWM published by the relative studies
## the RAP and CLIP PWM are computed by DREME with the parameters specified in the manuscript methods section
## For YBX3 in order to obtain the ACAHC motif, one background 4 mers (GAAC, masked to NNNN) needed to be added
## The RBFOX2 GMAUG motif is recomputed by weighing the abundance of GCAUG and GAAUG with the binding site length
## de novo motif discovery was performed only for RAPseq and eCLIPseq
## in figure 2A the logos for RBNS were drawn based on the logos published on the ENCODE Project website

```

### 0.4.1 eCLIP

```

#####
# eCLIP-seq data #####
# https://www.encodeproject.org
# HNRNPA1, accession ENCFF797GSK
# HNRNPC, accession ENCFF440R0Z
# PTBP1, accession ENCFF726SQU
# RBFOX2, accession ENCFF871NYM
# YBX3, accession ENCFF185OEI

# annotated data
PATH_eCLIP_ann <- "./Data/eCLIP/annotated/"
PEAKS_eCLIP_ann <- list.files(path = PATH_eCLIP_ann)
Names_eCLIP_ann <- unlist(str_split(PEAKS_eCLIP_ann, "\\.p"))[grep("txt", unlist(str_split(PEAKS_eCLIP_ann, "\\.p")))]
CLIPs <- list()
for (i in 1:length(Names_eCLIP_ann)){
  CLIPs[[i]] <- as.data.frame(read.table(paste(PATH_eCLIP_ann, PEAKS_eCLIP_ann[i], sep = ""), header = T))
}
names(CLIPs) <- Names_eCLIP_ann

# PFM
PATH_eCLIP_PFM <- "./Data/eCLIP/PFMs/"
PEAKS_eCLIP_PFM <- list.files(path = PATH_eCLIP_PFM)
Names_eCLIP <- unlist(str_split(PEAKS_eCLIP_PFM, "\\."))
eCLIPseq <- list()
for (i in 1:length(Names_eCLIP)){
  eCLIPseq[[i]] <- t(as.data.frame(read.table(paste(PATH_eCLIP_PFM, PEAKS_eCLIP_PFM[i], sep = ""), stringsAsFactors = F)))
  rownames(eCLIPseq[[i]]) <- c("A", "C", "G", "U")
  colnames(eCLIPseq[[i]]) <- NULL
}

names(eCLIPseq) <- Names_eCLIP
eCLIPseq$HNRNPA1 <- cbind(eCLIPseq$HNRNPA1, c(0,0,0,0), c(0,0,0,0), c(0,0,0,0))
eCLIPseq$HNRNPC <- cbind(eCLIPseq$HNRNPC, c(0,0,0,0), c(0,0,0,0))

```

```

eCLIPseq$PTBP1 <- cbind(eCLIPseq$PTBP1, c(0,0,0,0))
eCLIPseq$RBFOX2 <- cbind(eCLIPseq$RBFOX2, c(0,0,0,0), c(0,0,0,0))
eCLIPseq$YBX3 <- cbind(eCLIPseq$YBX3, c(0,0,0,0), c(0,0,0,0))

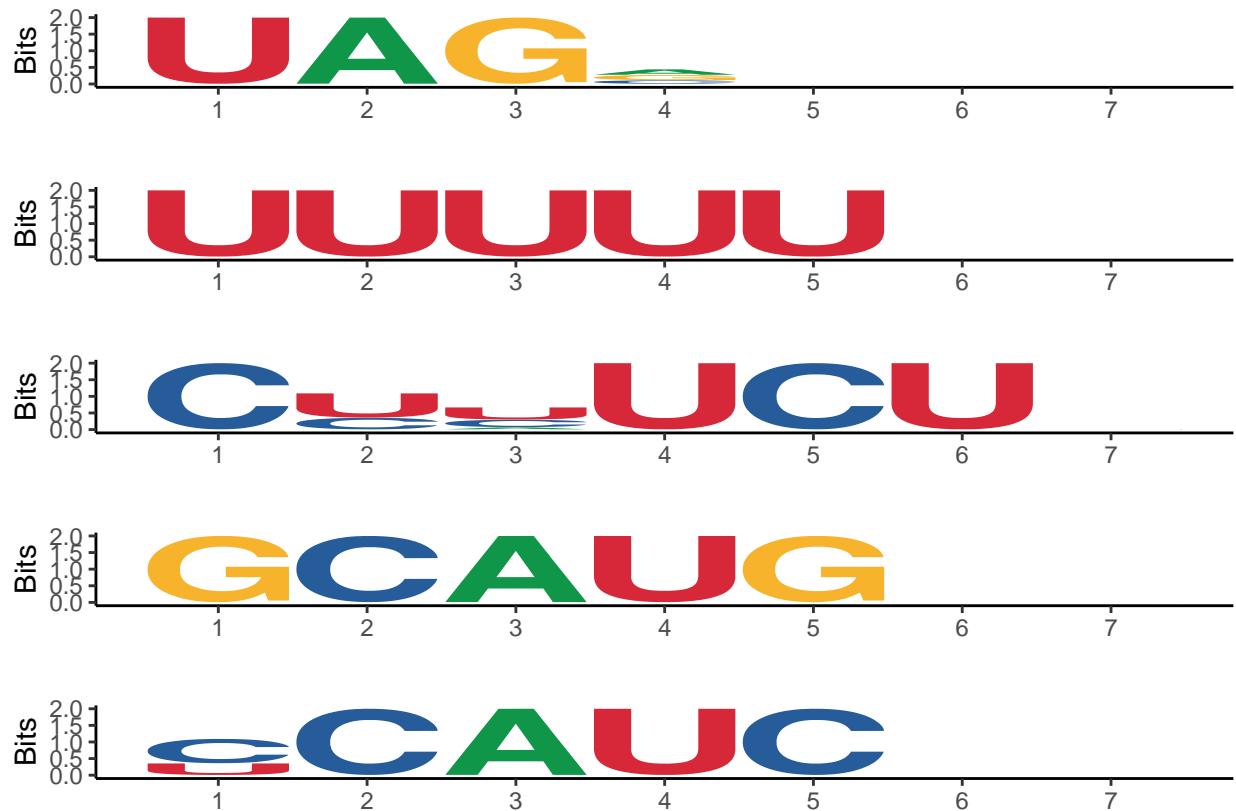
logos_eCLIPseq <- list()

for (i in 1:length(Names_eCLIP)){
  logos_eCLIPseq[[i]] <- ggplot() + geom_logo(eCLIPseq[[Names_eCLIP[i]]], font="helvetica_bold") + theme_minimal()
}

## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## i The deprecated feature was likely used in the ggseqlogo package.
## Please report the issue at <https://github.com/omarwagih/ggseqlogo/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

names(logos_eCLIPseq) <- Names_eCLIP
eCLIP_logo <- ggarrange(logos_eCLIPseq$HNRNPA1, logos_eCLIPseq$HNRNPC, logos_eCLIPseq$PTBP1, logos_eCLIPseq$YBX3)
eCLIP_logo

```



#### 0.4.2 RNA Compete

```

#####
# Ref: A compendium of RNA-binding motifs for decoding gene regulation
# Link: https://hugheslab.ccbr.utoronto.ca/supplementary-data/RNAcompete_eukarya/
# HNRNPA1: https://hugheslab.ccbr.utoronto.ca/supplementary-data/RNAcompete_eukarya/Experiment_reports/
# HNRNPC: https://hugheslab.ccbr.utoronto.ca/supplementary-data/RNAcompete_eukarya/Experiment_reports/
# PTBP1: https://hugheslab.ccbr.utoronto.ca/supplementary-data/RNAcompete_eukarya/Experiment_reports/
# https://hugheslab.ccbr.utoronto.ca/supplementary-data/RNAcompete_eukarya/Experiment_reports

PATH_RNACompete <- "./Data/RNAcompete/"
PEAKS_RNACompete <- list.files(path = PATH_RNACompete)
Names_RNACompete <- unlist(str_split(PEAKS_RNACompete, "\\.")) [grep("txt", unlist(str_split(PEAKS_RNACompete, "\\.")))]
RNACompete <- list()
for (i in 1:length(Names_RNACompete)){
  RNACompete[[i]] <- as.data.frame(read.table(paste(PATH_RNACompete, PEAKS_RNACompete[i], sep = ""), stringsAsFactors = FALSE))
  RNACompete[[i]] <- t(RNACompete[[i]][,-1])
}
names(RNACompete) <- Names_RNACompete

logos_plots_RNACompete <- list()
for (i in 1:length(Names_RNACompete)){
  logos_plots_RNACompete[[i]] <- ggplot() + geom_logo(RNACompete[[Names_RNACompete[i]]]), font="helvetica")
}
names(logos_plots_RNACompete) <- Names_RNACompete
RNA_Compete_logo <- ggarrange(logos_plots_RNACompete$HNRNPA1, logos_plots_RNACompete$HNRNPC, logos_plots_RNACompete$PTBP1)

```



### 0.4.3 RBNS

```
# Ref: Sequence, Structure, and Context Preferences of Human RNA Binding Proteins
# Link: https://ars.els-cdn.com/content/image/1-s2.0-S1097276518303514-mmc4.xlsx
# Link: https://www.encodeproject.org/encore-matrix/?type=Experiment&status=released&internal_tags=ENCODE
# HNRNPC: https://www.encodeproject.org/experiments/ENCSR569UIU/
# https://www.encodeproject.org/documents/35943eb4-837d-4f96-a705-46eec6838f5d/@@download/attach
# RBFOX2: https://www.encodeproject.org/experiments/ENCSR441HLP/
# https://www.encodeproject.org/documents/2403659a-974e-4312-900f-45d0c8e3566d/@@download/attach
```

### 0.4.4 RAPSeq

```
##### RAP SEQ #####
PATH_RAP_ann <- "./Data/RAPseq/annotated/"
PEAKS_RAP_ann <- list.files(path = PATH_RAP_ann)
Names_RAP <- unlist(str_split(PEAKS_RAP_ann, "\\\\.p"))[grep("txt", unlist(str_split(PEAKS_RAP_ann, "\\\\.p")))
RAPS <- list()
for (i in 1:length(Names_RAP)){
  print(paste(PATH_RAP_ann, PEAKS_RAP_ann[i], sep = ""))
  RAPs[[i]] <- as.data.frame(read.table(paste(PATH_RAP_ann, PEAKS_RAP_ann[i], sep = ""), header = T, stringsAsFactors = F))
}
## [1] "./Data/RAPseq/annotated/HNRNPA1.peaks.txt"
## [1] "./Data/RAPseq/annotated/HNRNPC.peaks.txt"
## [1] "./Data/RAPseq/annotated/hsHuRHUMAN.peaks.txt"
## [1] "./Data/RAPseq/annotated/HuRPTBP1.peaks.txt"
## [1] "./Data/RAPseq/annotated/IGF2BP1.peaks.txt"
## [1] "./Data/RAPseq/annotated/PTBP1.peaks.txt"
## [1] "./Data/RAPseq/annotated/RBFOX2.peaks.txt"
## [1] "./Data/RAPseq/annotated/YBX3.peaks.txt"
names(RAPs) <- Names_RAP

### A C G U #####
#GCATG <- RAPs[["RBFOX2"]][grep("GAATG", str_sub(RAPs[["RBFOX2"]]$positive_fa, 80, 120), invert=T), ]
GCATG <- RAPs[["RBFOX2"]][grep("GCATG", str_sub(RAPs[["RBFOX2"]]$positive_fa, 80, 120)), "BS"]
GCATG <- rep("GCATG", round(sqrt(length(GCATG))*median(GCATG)) )

#GAATG <- RAPs[["RBFOX2"]][grep("GCATG", str_sub(RAPs[["RBFOX2"]]$positive_fa, 80, 120), invert=T), ]
GAATG <- RAPs[["RBFOX2"]][grep("GAATG", str_sub(RAPs[["RBFOX2"]]$positive_fa, 80, 120)), "BS"]
GAATG <- rep("GAATG", round(sqrt(length(GAATG))*median(GAATG)) )
GMATG <- c(GCATG, GAATG)
GMATG <- DNAStringSet(GMATG)
GMATG <- consensusMatrix(DNAStringSet(GMATG))[1:4,]
rownames(GMATG) <- c("A", "C", "G", "U")
GMAUG <- GMATG
GMAUG <- cbind(GMAUG, c(0, 0, 0, 0), c(0, 0, 0, 0))

GMAUG_logo <- ggplot() + geom_logo(GMAUG, font="helvetica_bold") + theme_classic() + ylim(0, 2)

PATH_RAP_PFM <- "./Data/RAPseq/PFMs/"
PEAKS_RAP_PFM <- list.files(path = PATH_RAP_PFM)
Names_RAP_PFM <- unlist(str_split(PEAKS_RAP_PFM, "\\\\."))[grep("txt", unlist(str_split(PEAKS_RAP_PFM, "\\\\.p")))]
```

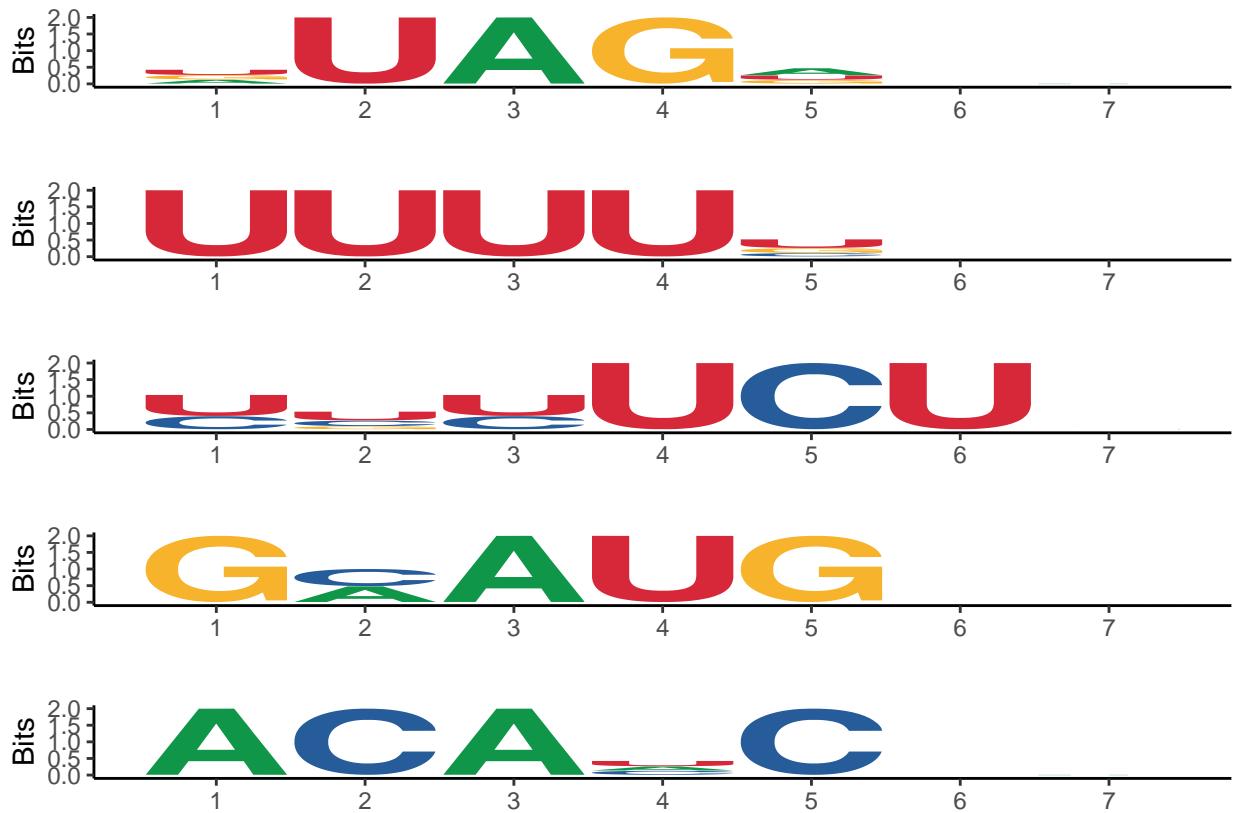
```

RAPseq <- list()

for (i in 1:length(Names_RAP_PFMs)){
  RAPseq[[i]] <- t(as.data.frame(read.table(paste(PATH_RAP_PFM, PEAKS_RAP_PFM[i], sep = ""), stringsAsFactors = TRUE)))
  rownames(RAPseq[[i]]) <- c("A", "C", "G", "U")
  colnames(RAPseq[[i]]) <- NULL
}

names(RAPseq) <- Names_RAP_PFM
RAPseq$HNRNPA1 <- cbind(RAPseq$HNRNPA1, c(0,0,0,0), c(0,0,0,0))
RAPseq$HNRNPC <- cbind(RAPseq$HNRNPC, c(0,0,0,0), c(0,0,0,0))
RAPseq$PTBP1 <- cbind(RAPseq$PTBP1, c(0,0,0,0))
RAPseq$YBX3 <- cbind(RAPseq$YBX3, c(0,0,0,0), c(0,0,0,0))
logos_RAPseq <- list()
for (i in 1:length(Names_RAP_PFM)){
  logos_RAPseq[[i]] <- ggplot() + geom_logo(RAPseq[[Names_RAP_PFM[i]]], font="helvetica_bold") + theme_minimal()
}
names(logos_RAPseq) <- Names_RAP_PFM
RAP_logo <- ggarrange(logos_RAPseq$HNRNPA1, logos_RAPseq$HNRNPC, logos_RAPseq$PTBP1, GMAUG_logo, logos_RAPseq$YBX3)
RAP_logo

```



## 0.5 Figures settings

### 0.5.1 Figure 2A

```
# RAP_logo, eCLIP_logo, RNA_Compete_logo and HTRSELEX_logo
# RBNS data were from ENCODE Project
p_motifLOGO <- ggarrange(RAP_logo,eCLIP_logo, RNA_Compete_logo, NULL, ncol=4)
ggsave(filename = "./Figure/Figure2A_motifLogo.pdf", plot = p_motifLOGO, width = 11.5, height = 7.8)
```

### 0.5.2 Figure 2B

```
# uracil content
RAPS[["HNRNPC"]]$Us <- str_count(str_sub(RAPS[["HNRNPC"]])$positive_fa, 80, 120), "T") / (120-80)
CLIPs[["HNRNPC"]]$Us <- str_count(str_sub(CLIPs[["HNRNPC"]])$positive_fa, 43, 83), "T") / (83-43)

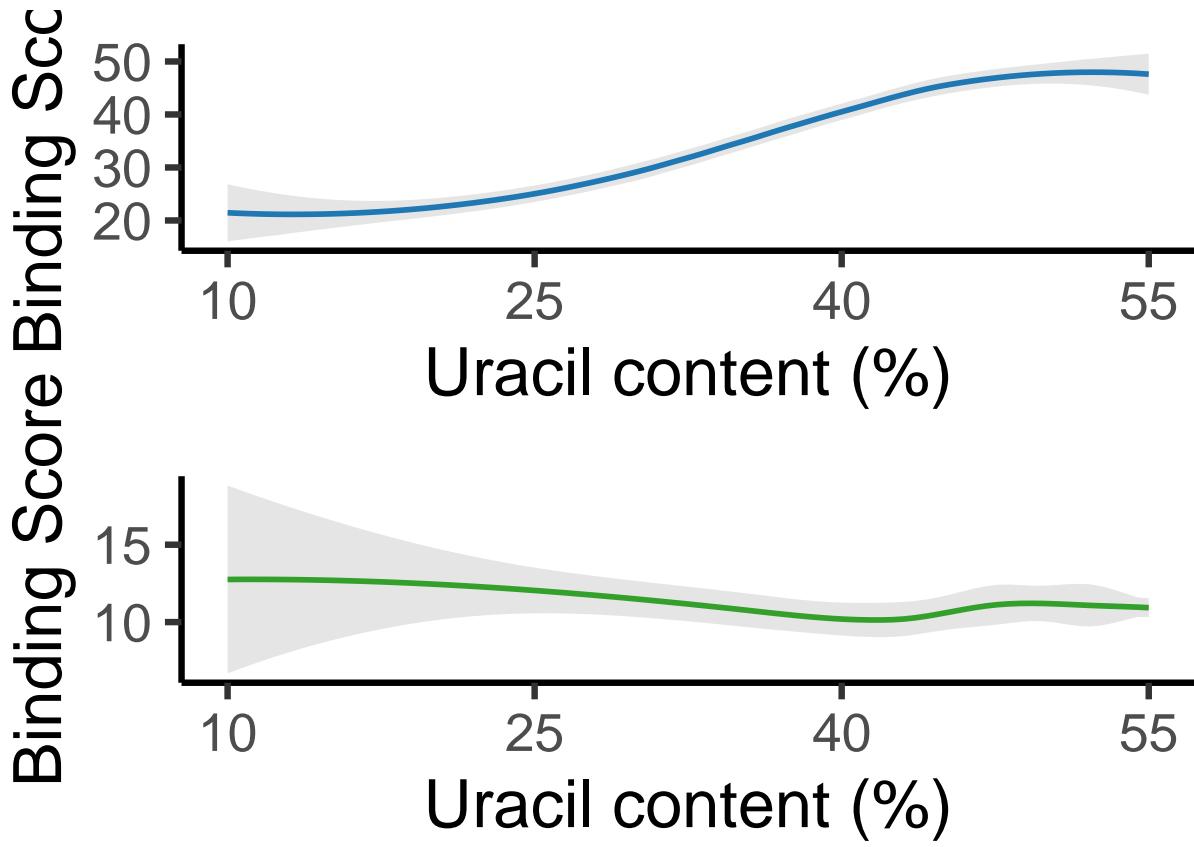
#
RAP_Us_data <- RAPS[["HNRNPC"]][,c("Us", "BS", "Mean_FCI")]
colnames(RAP_Us_data) <- c("Us", "BS", "MeanFC")
RAP_Us_data$Assay <- rep("RAP", nrow(RAP_Us_data))
CLIP_Us_data <- CLIPs[["HNRNPC"]][,c("Us", "BS", "Mean_FC")]
colnames(CLIP_Us_data) <- c("Us", "BS", "MeanFC")
CLIP_Us_data$Assay <- rep("CLIP", nrow(CLIP_Us_data))

HNRNPC_Us_data <- rbind(RAP_Us_data, CLIP_Us_data)
HNRNPC_Us_data$Us <- HNRNPC_Us_data$Us*100
HNRNPC_Us_data$Us[HNRNPC_Us_data$Us < 10] <- 10
HNRNPC_Us_data$Us[HNRNPC_Us_data$Us > 55] <- 55

fig_bindingScore_rap <- ggplot2::ggplot(data=HNRNPC_Us_data[HNRNPC_Us_data$Assay == "RAP",], aes(x=Us,
  stat_smooth(method="loess", color="#1f78b4", fill=greys[6], level = 0.99) +
  theme_classic2(base_size = 25) +
  ylab("Binding Score") +
  xlab("Uracil content (%)") +
  scale_x_continuous(breaks = c(10, 25, 40, 55))

fig_bindingScore_clip <- ggplot2::ggplot(data=HNRNPC_Us_data[HNRNPC_Us_data$Assay == "CLIP",], aes(x=Us,
  stat_smooth(method="loess", color="#33a02c", fill=greys[6], level = 0.99) +
  theme_classic2(base_size = 25) +
  ylab("Binding Score") +
  xlab("Uracil content (%)") +
  scale_x_continuous(breaks = c(10, 25, 40, 55))

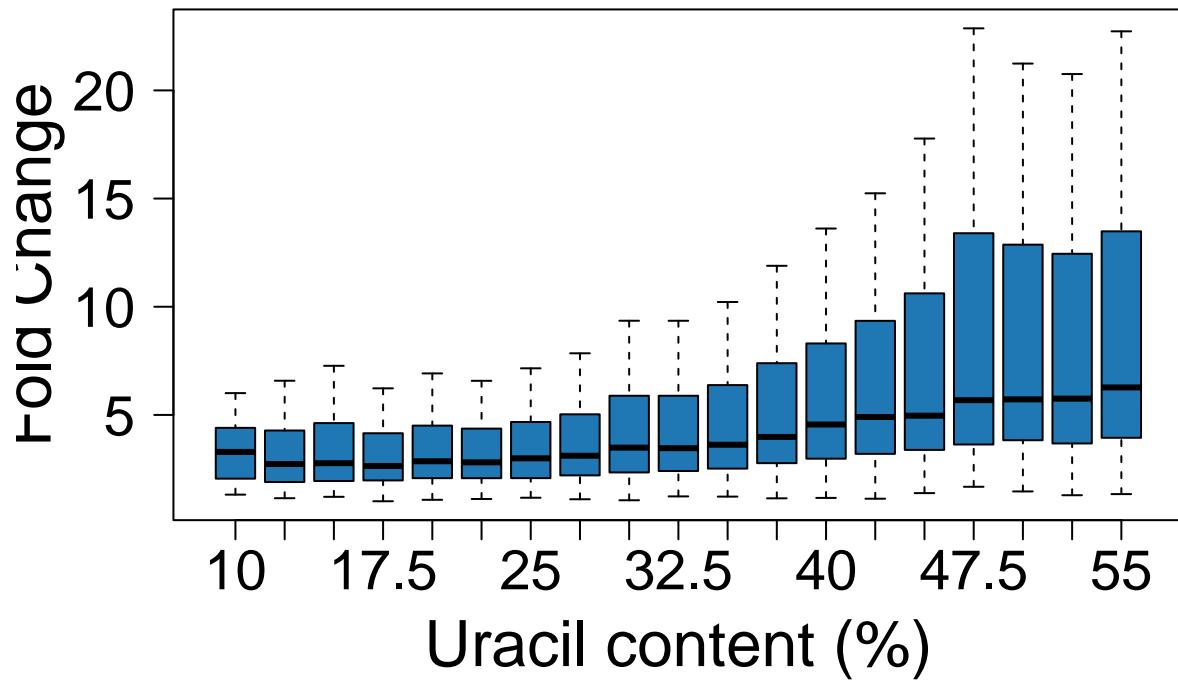
p_bindingScore_Us <- ggarrange(fig_bindingScore_rap, fig_bindingScore_clip, ncol=1)
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
p_bindingScore_Us
```



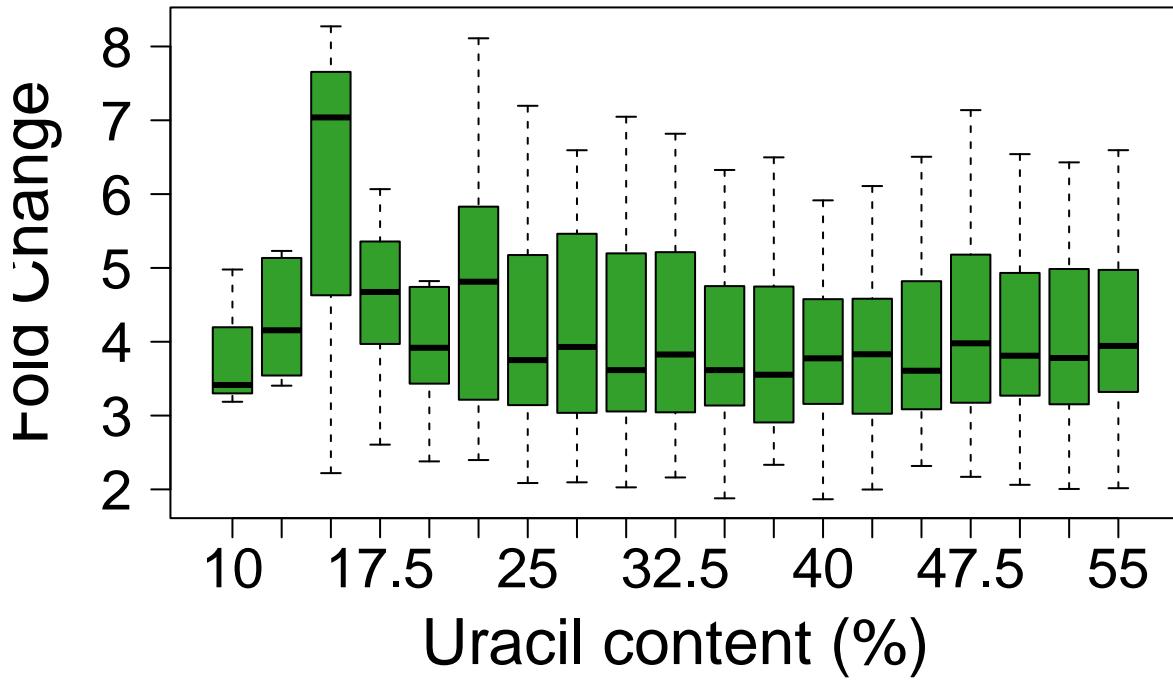
```

ggsave(filename = "./Figure/Figure2B_BindingScoreUracilContent_LOSSES.pdf", plot = p_bindingScore_Us, 
#pdf("./Figure/Figure2B_UracilContentBoxplot_RAP.pdf", width = 11.5, height = 7.8)
boxplot(data=HNRNPC_Us_data[HNRNPC_Us_data$Assay == "RAP",], MeanFC~Us, outline=F, range=1, notch=F, colo

```



```
#dev.off()  
  
#pdf("./Figure/Figure2B_UracilContentBoxplot_CLIP.pdf", width = 11.5, height = 7.8)  
boxplot(data=HNRPUs_data[HNRPUs_data$Assay == "CLIP",], MeanFC~Us, outline=F, range=1, notch=F, c
```



```
#dev.off()
```

### 0.5.3 Figure 2C

```
## DeepTools count matrices (bin size = 10nt) are read in
## DeepTools was used with the following command to generate the per nt count matrices
## computeMatrix reference-point -S rep1.bw -R rep1.bed -a 100 -b 100 -bs 10
## RAPseq
PATH_RAP_Coverage <- "./Data/RAPseq/Coverage/10nt/"
PEAKS_RAP_Coverage <- list.files(path = PATH_RAP_Coverage)
Names_RAP_Coverage <- unlist(str_split(PEAKS_RAP_Coverage, "\\\\.matr"))[grep("txt", unlist(str_split(PEAKS_RAP_Coverage, "\\\\.matr")))]
RAPseq_Coverage <- list()

for (i in 1:length(Names_RAP_Coverage)){
  RAPseq_Coverage[[i]] <- as.data.frame(read.table(paste(PATH_RAP_Coverage, PEAKS_RAP_Coverage[i], sep = "."))
}
names(RAPseq_Coverage) <- Names_RAP_Coverage

#par(mfrow=c(4,5), bty="n")
Names_protein <- c("HNRNPA1", "HNRNPC", "PTBP1", "RBFOX2", "YBX3")
# coverage normalized RPM
for (i in 1:length(Names_protein)) {

  rep1_name <- paste (Names_protein[i], ".rep1", sep = "")
```

```

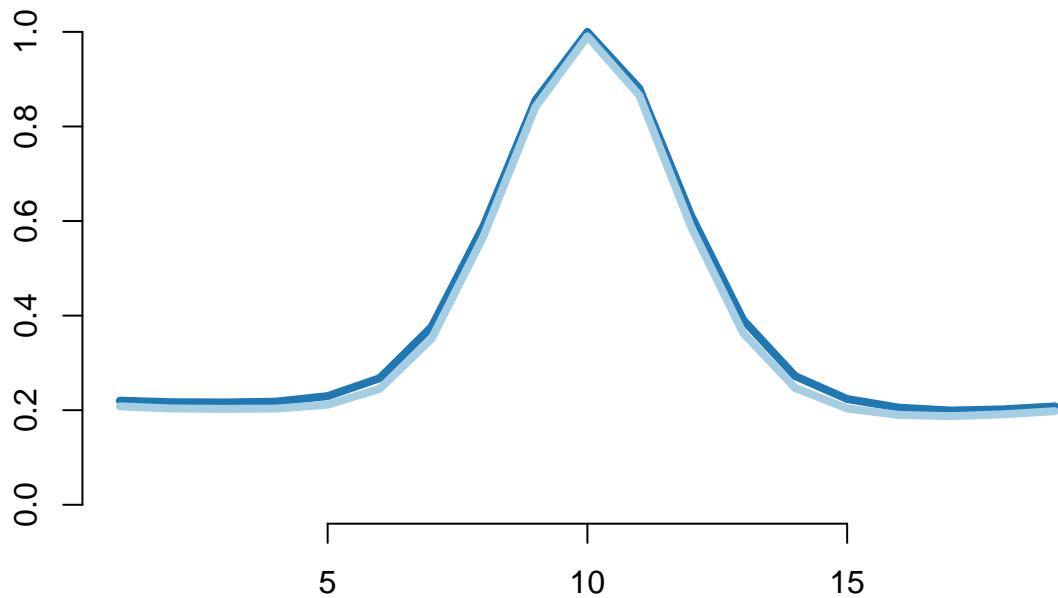
rep2_name <- paste (Names_protein[i] , ".rep2" , sep="")

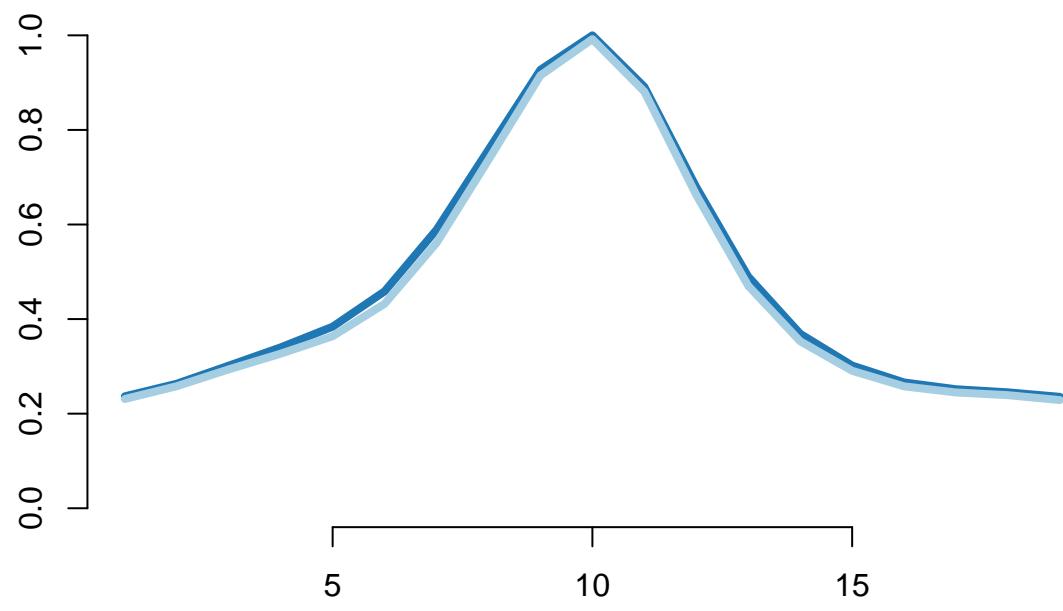
mean_rep1 <- rollapply(colMeans(as.matrix(RAPseq_Coverage[[rep1_name]])), width=2, FUN = mean)
mean_rep2 <- rollapply(colMeans(as.matrix(RAPseq_Coverage[[rep2_name]])), width=2, FUN = mean)

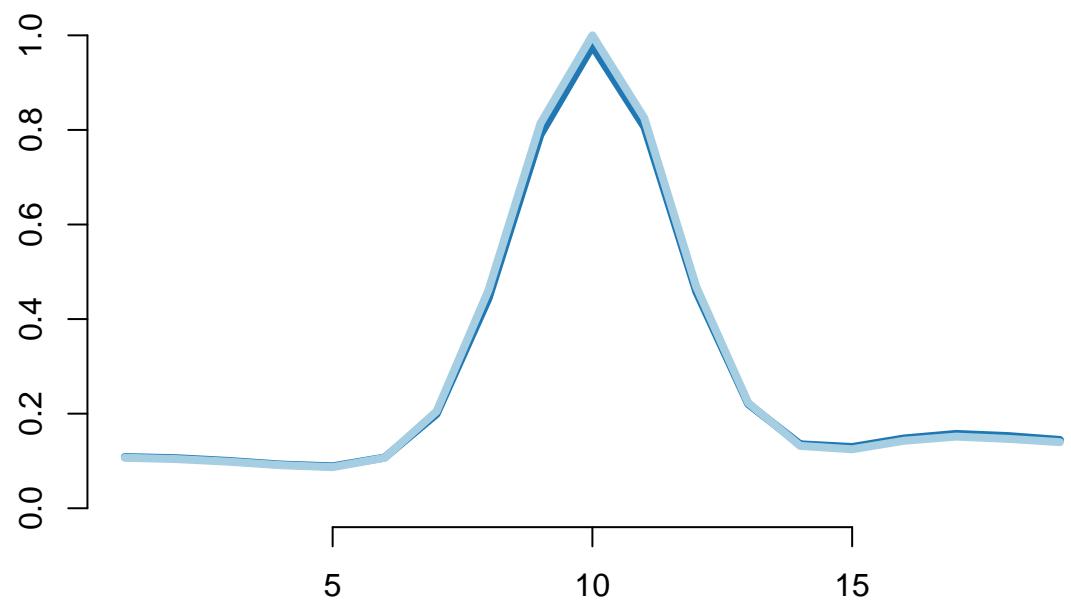
normalize_rep1 <- mean_rep1/max(c(mean_rep1,mean_rep2))
normalize_rep2 <- mean_rep2/max(c(mean_rep1,mean_rep2))

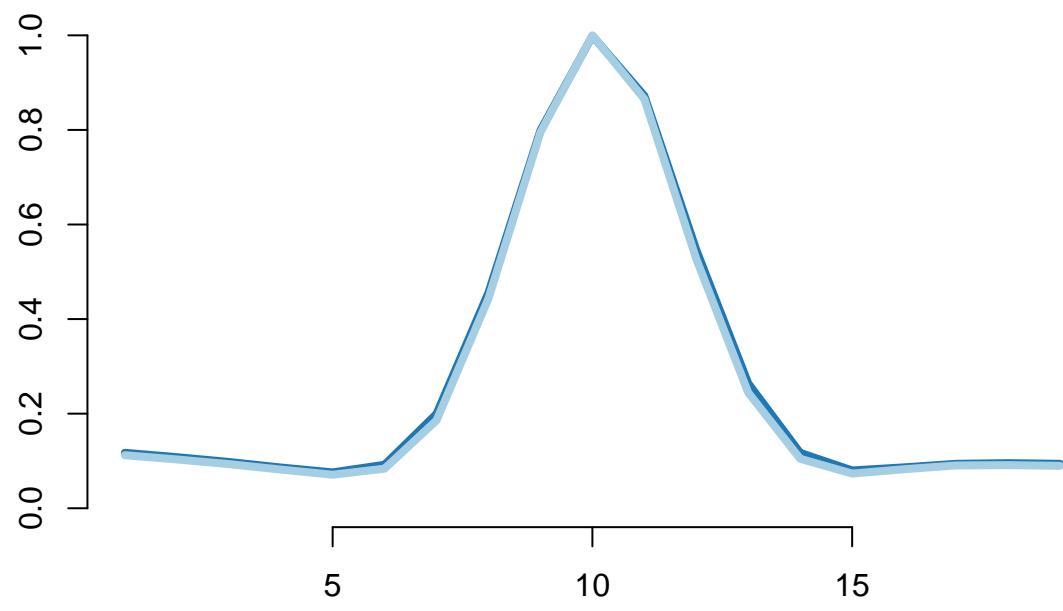
plot(normalize_rep1, type="l", col="#1f78b4", lwd=4, ylab=NA, xlab=NA, ylim=c(0,1), las=1, axes=FALSE)
axis(side=1, labels=T)
axis(side=2, labels=T)
points(normalize_rep2,type="l",col="#a6cee3", lwd=4)
}

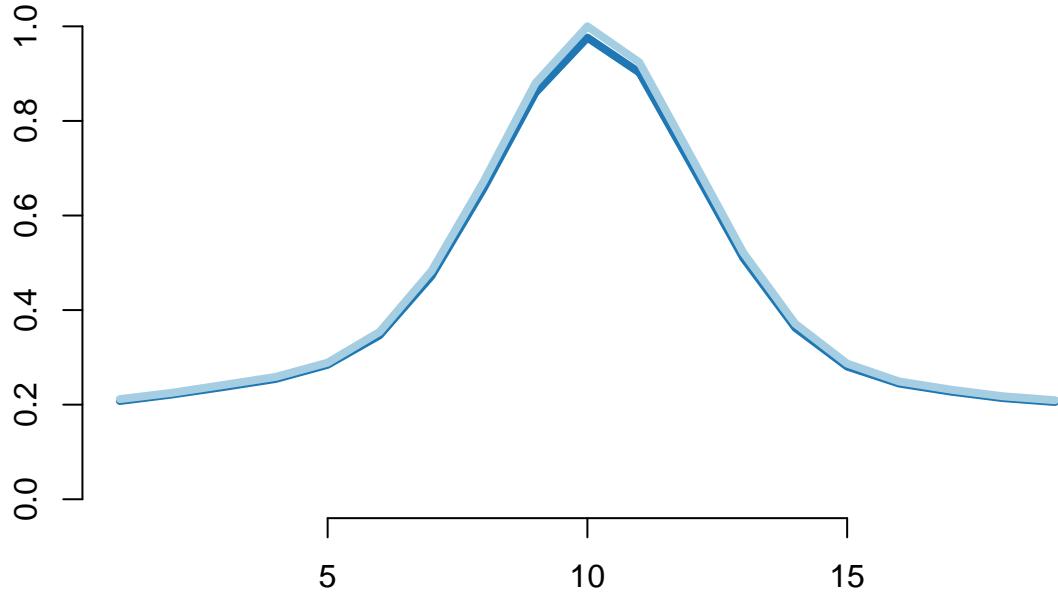
```











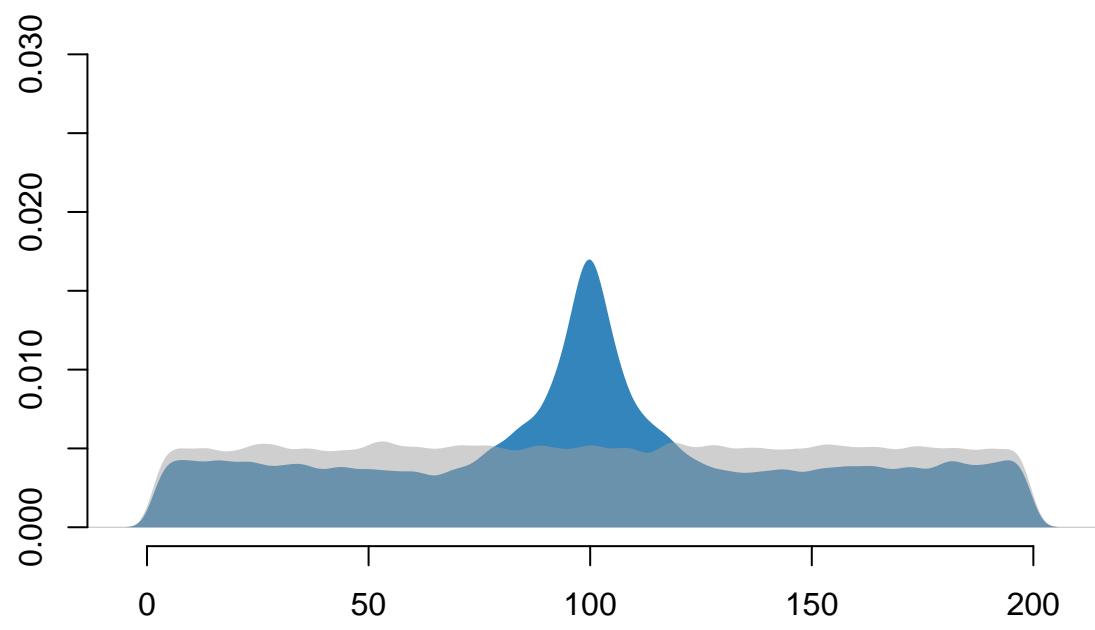
```

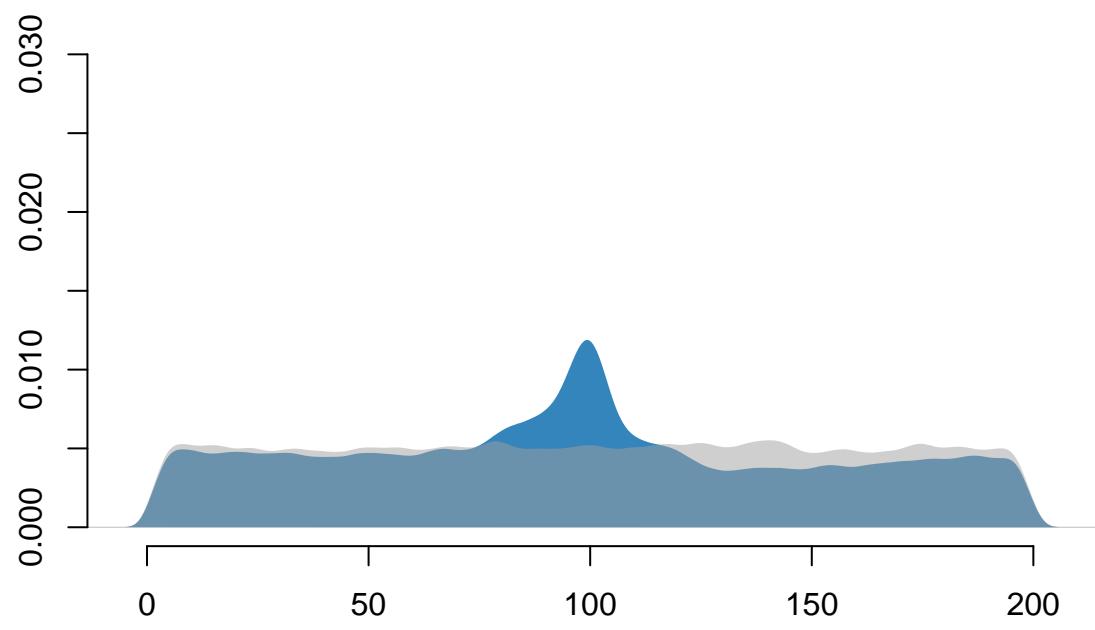
# motif density
RAPs_motifdensity <- RAPs
RAPs_motifdensity[["PTBP1"]]\$positive_fa <- gsub("T", "Y", RAPs_motifdensity[["PTBP1"]]\$positive_fa)
RAPs_motifdensity[["PTBP1"]]\$positive_fa <- gsub("C", "Y", RAPs_motifdensity[["PTBP1"]]\$positive_fa)
RAPs_motifdensity[["PTBP1"]]\$negative_fa <- gsub("T", "Y", RAPs_motifdensity[["PTBP1"]]\$negative_fa)
RAPs_motifdensity[["PTBP1"]]\$negative_fa <- gsub("C", "Y", RAPs_motifdensity[["PTBP1"]]\$negative_fa)

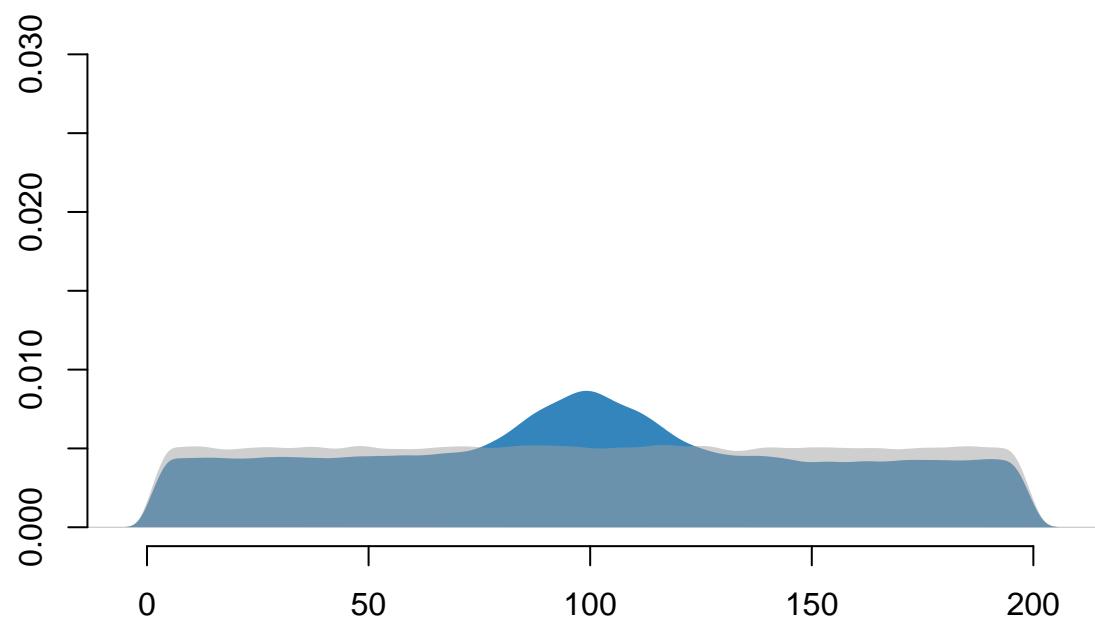
Motifs <- c("TAG", "TTTT", "YYYY", "GCATG|GAATG", "AACATC|AACACAC|AACACC")

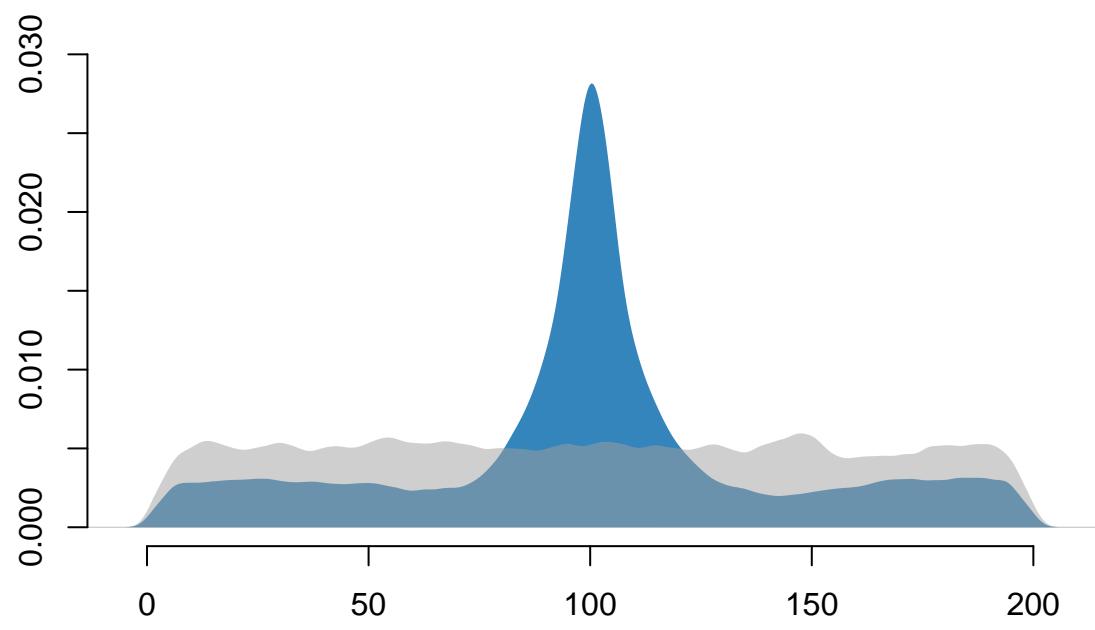
for (i in 1:length(Names_protein)){
  signal <- density(unlist(str_locate_all(RAPs_motifdensity[[Names_protein[i]]]\$positive_fa,Motifs[i])))
  noise <- density(unlist(str_locate_all(RAPs_motifdensity[[Names_protein[i]]]\$negative_fa,Motifs[i])))
  plot(signal, col=NA, main=NA, xlab=NA, ylab=NA, ylim=c(0,0.03), axes=FALSE)
  axis(side=1, labels=T)
  axis(side=2, labels=T)
  polygon(signal, col=alpha("#1f78b4", 0.9),border = NA)
  polygon(noise, col=alpha("#AOAOAO", 0.5),border = NA)
}

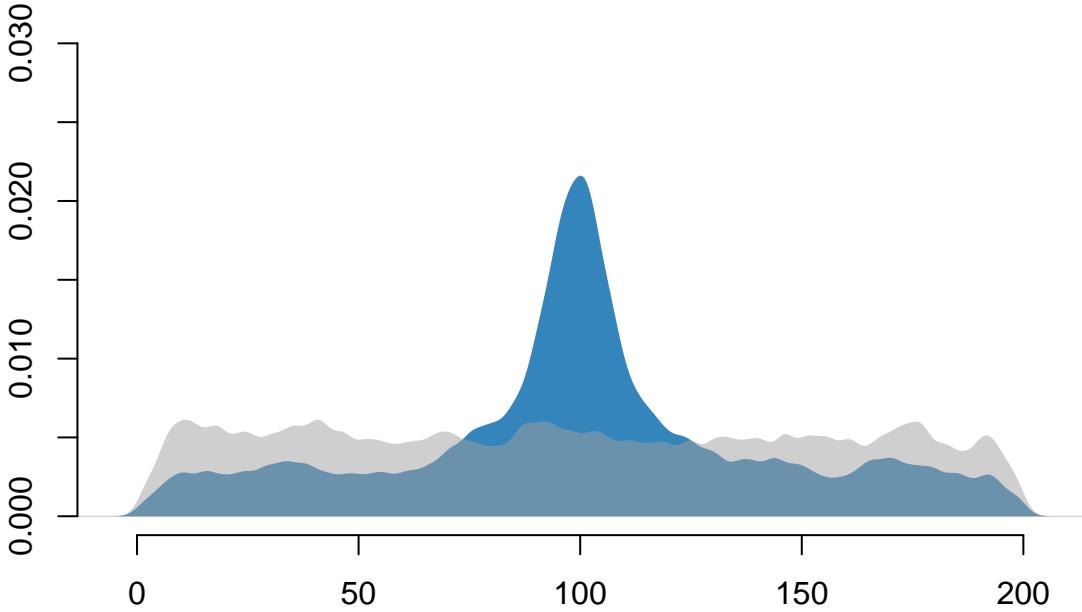
```











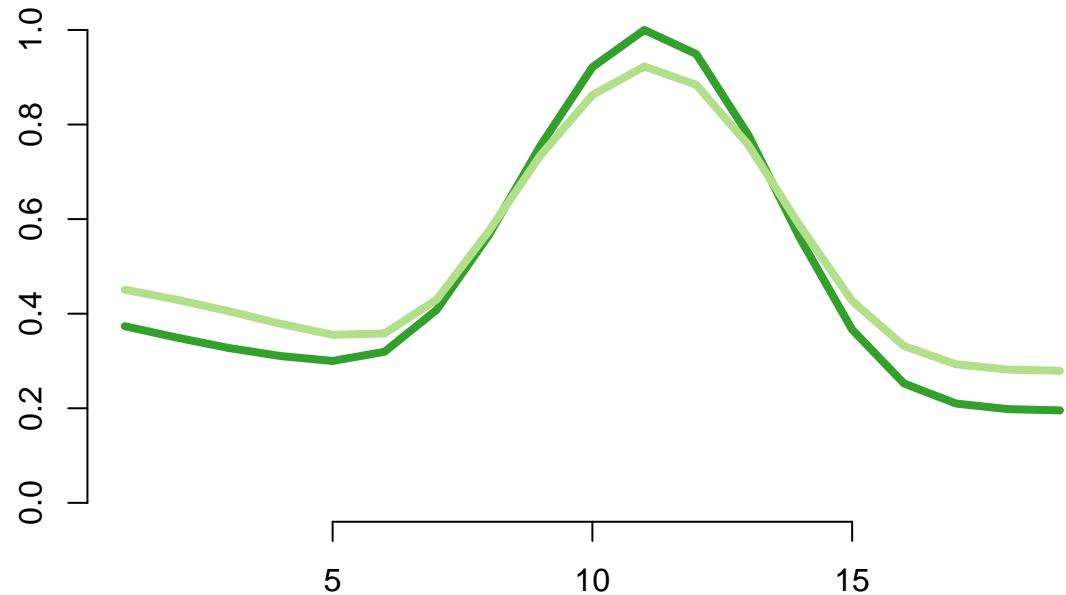
```

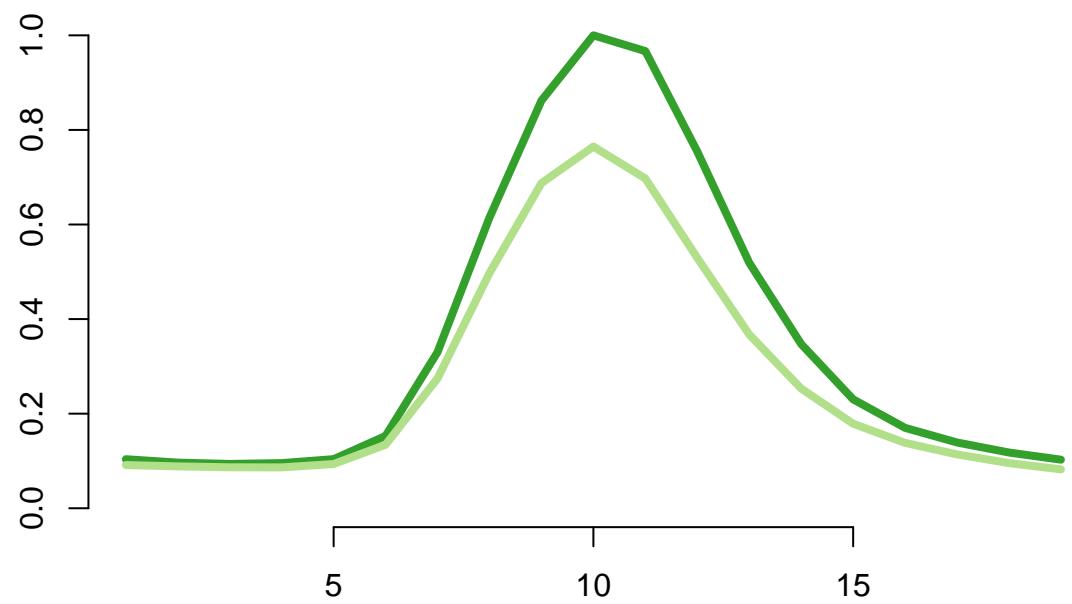
# eCLIP
## DeepTools count matrices (bin size = 10nt) are read in
## DeepTools was used with the following command to generate the per nt count matrices
## computeMatrix reference-point -S rep1.bw -R rep1.bed -a 100 -b 100 -bs 10
PATH_eCLIP_Coverage <- "./Data/eCLIP/Coverage/10nt/"
PEAKS_eCLIP_Coverage <- list.files(path = PATH_eCLIP_Coverage)
Names_eCLIP_Coverage <- unlist(str_split(PEAKS_eCLIP_Coverage, "\\\\.matr"))[grep("txt", unlist(str_split(PEAKS_eCLIP_Coverage, "\\\\.matr")))]
CLIPseq_Coverage <- list()
for (i in 1:length(Names_eCLIP_Coverage)){
  CLIPseq_Coverage[[i]] <- as.data.frame(read.table(paste(PATH_eCLIP_Coverage, PEAKS_eCLIP_Coverage[i], sep = "."))
}
names(CLIPseq_Coverage) <- Names_eCLIP_Coverage
Names_eCLIP_Coverage <- c("HNRNPA1_eCLIP", "HNRNPC_eCLIP", "PTBP1_eCLIP", "RBFOX2_eCLIP", "YBX3_eCLIP")
# coverage normalized RPM
for (i in 1:length(Names_eCLIP_Coverage)){
  rep1_name <- paste(Names_eCLIP_Coverage[i], ".rep1", sep = "")
  rep2_name <- paste(Names_eCLIP_Coverage[i], ".rep2", sep = "")

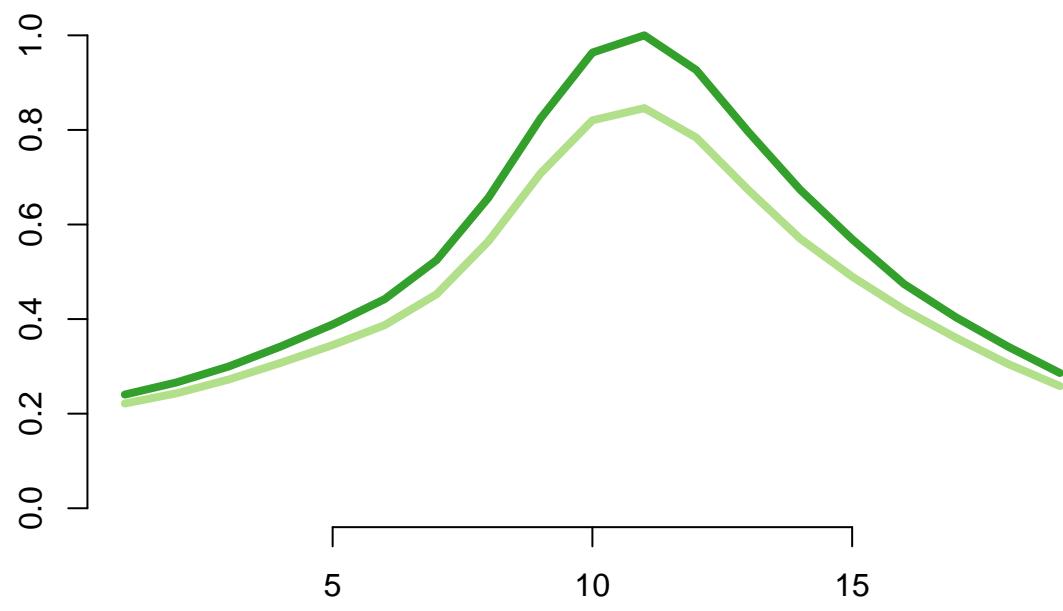
  rep1_mean <- rollapply(colMeans(as.matrix(CLIPseq_Coverage[[rep1_name]])), width=2, FUN = mean)
  rep2_mean <- rollapply(colMeans(as.matrix(CLIPseq_Coverage[[rep2_name]])), width=2, FUN = mean)

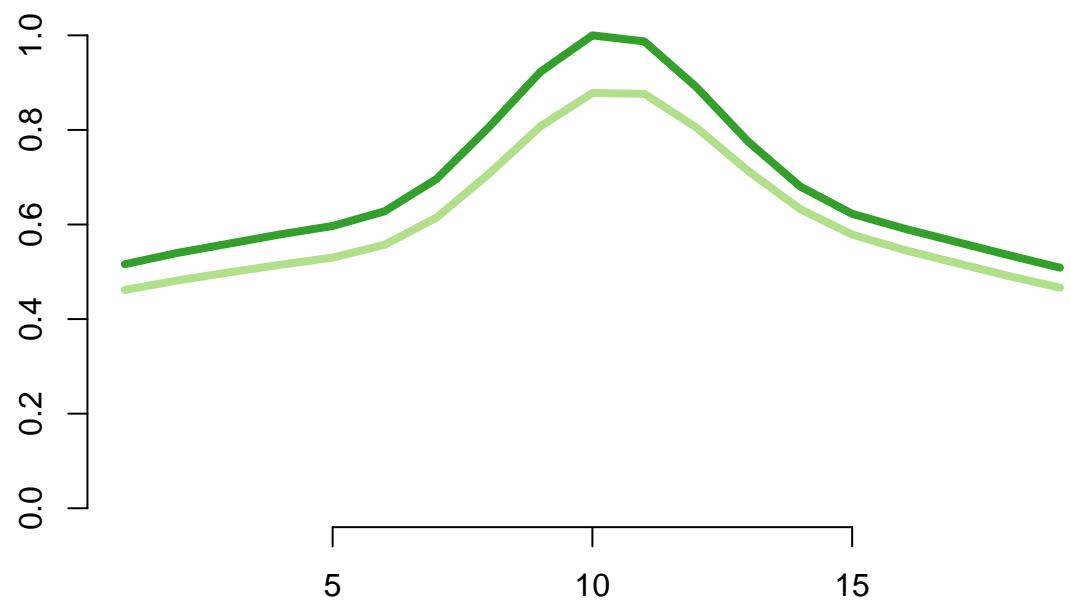
  x <- rep1_mean/max(c(rep1_mean, rep2_mean))
  y <- rep2_mean/max(c(rep1_mean, rep2_mean))
}
```

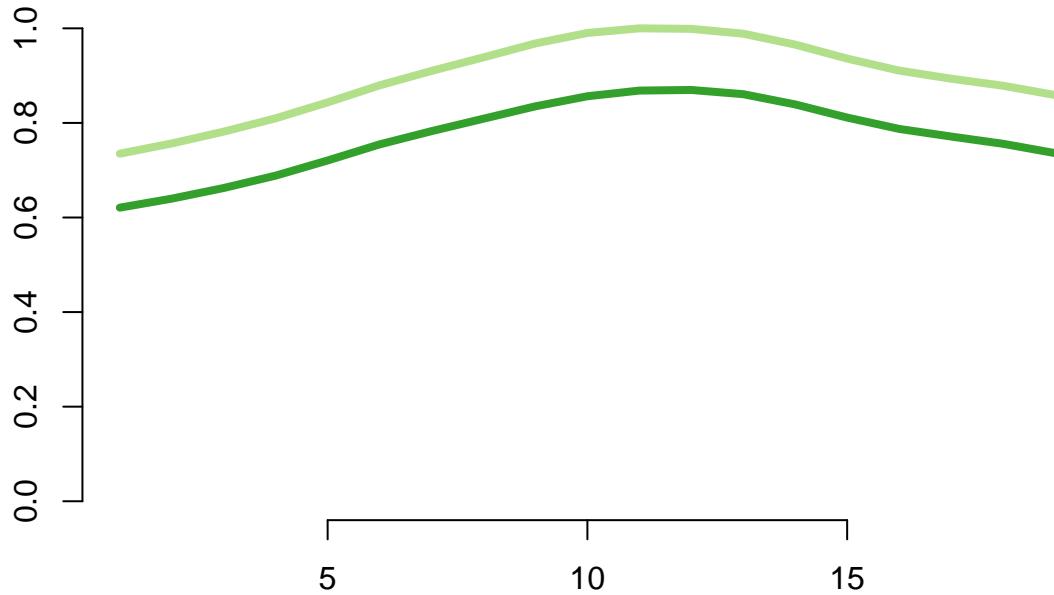
```
plot(x,type="l",col="#33a02c", lwd=4, ylab=NA, xlab=NA, ylim=c(0,1), las=1, axes=FALSE)
axis(side=1, labels=T)
axis(side=2, labels=T)
points(y,type="l",col="#b2df8a", lwd=4)
}
```











```

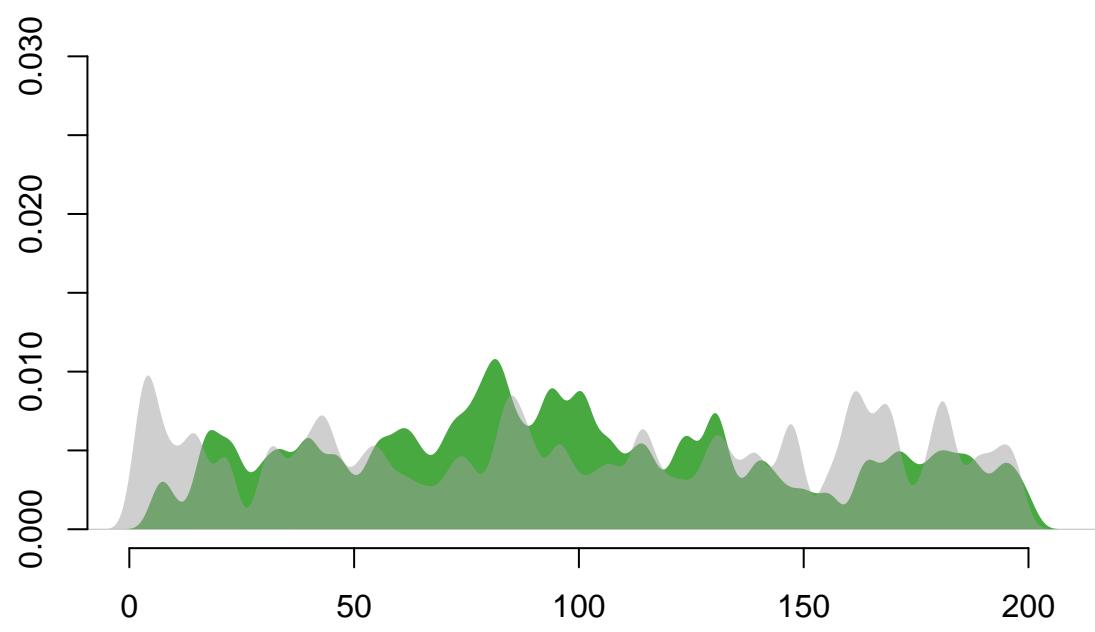
# motif density
CLIPs[["PTBP1"]]$positive_fa <- gsub("T", "Y", CLIPs[["PTBP1"]]$positive_fa)
CLIPs[["PTBP1"]]$positive_fa <- gsub("C", "Y", CLIPs[["PTBP1"]]$positive_fa)
CLIPs[["PTBP1"]]$negative_fa <- gsub("T", "Y", CLIPs[["PTBP1"]]$negative_fa)
CLIPs[["PTBP1"]]$negative_fa <- gsub("C", "Y", CLIPs[["PTBP1"]]$negative_fa)

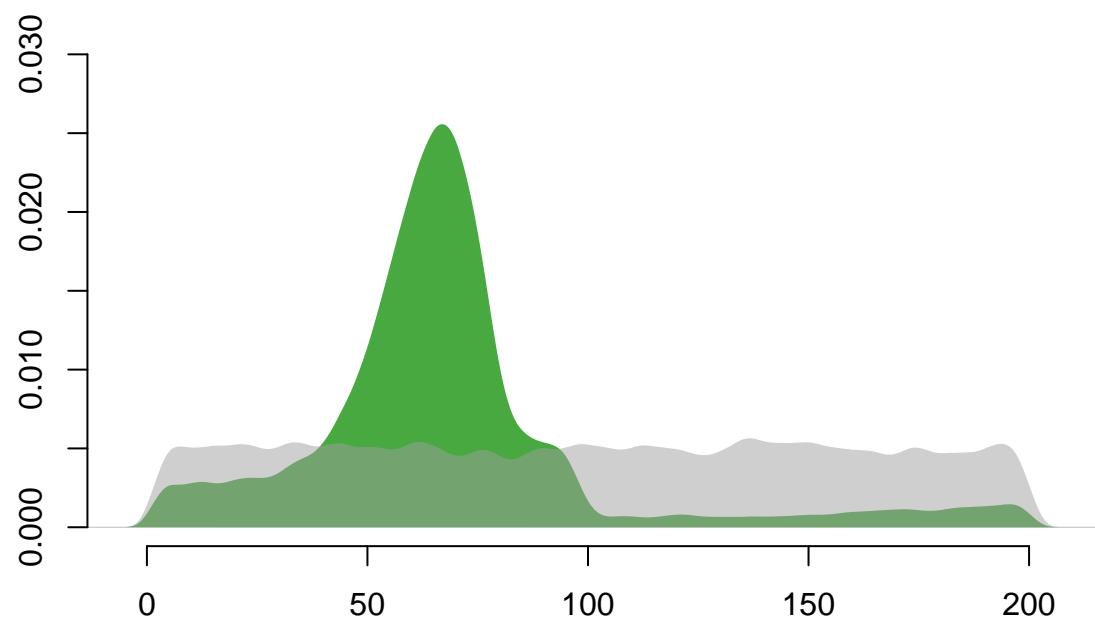
Motifs <- c("TAG", "TTTT", "YYYY", "GCATG", "CCATC|TCATC")
Names_protein <- c("HNRNPA1", "HNRNPC", "PTBP1", "RBFOX2", "YBX3")
for (i in 1:length(Names_protein)) {

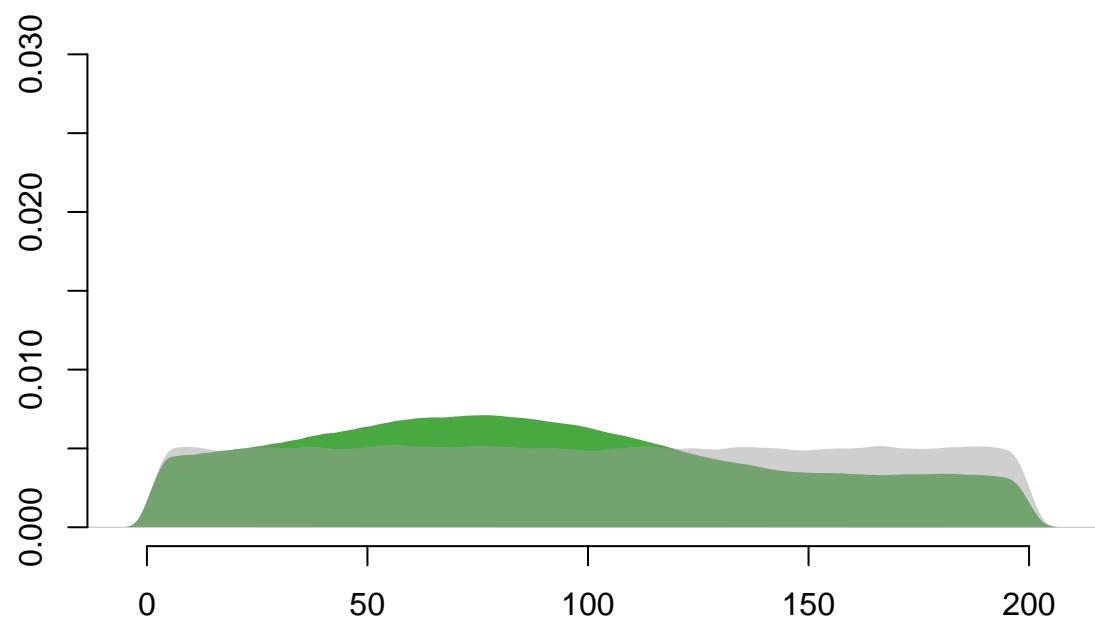
  signal <- density(unlist(str_locate_all(CLIPs[[Names_protein[i]]]$positive_fa,Motifs[i])),bw=2)
  noise <- density(unlist(str_locate_all(CLIPs[[Names_protein[i]]]$negative_fa,Motifs[i])),bw=2)

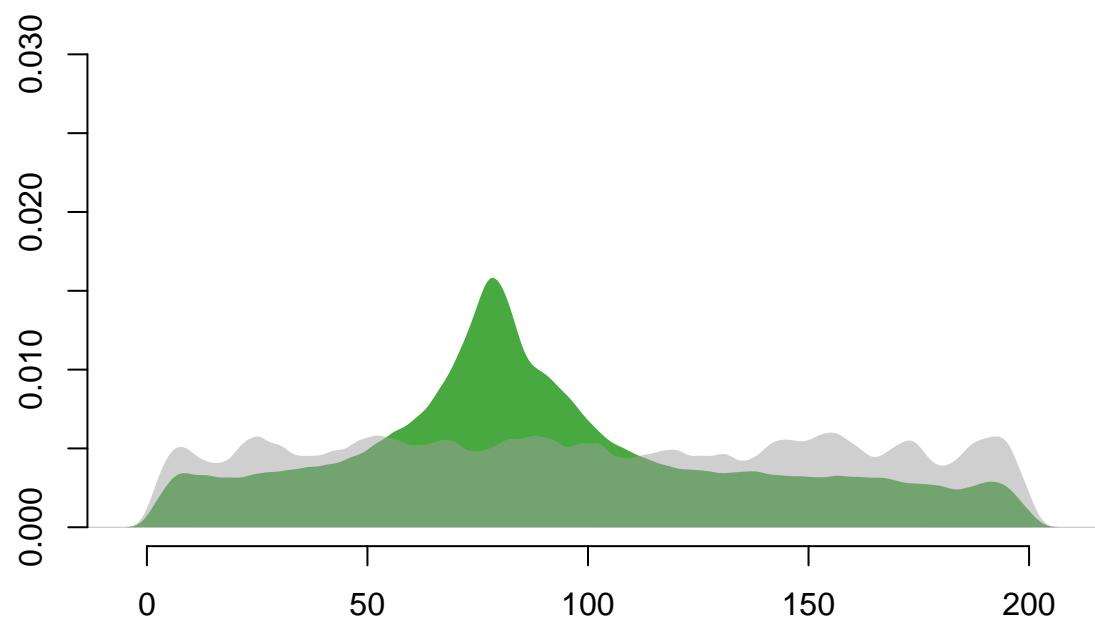
  plot(signal, col=NA, main=NA, xlab=NA, ylab=NA, ylim=c(0,0.03), axes=FALSE)
  axis(side=1, labels=T)
  axis(side=2, labels=T)
  polygon(signal, col=alpha("#33a02c", 0.9),border = NA)
  polygon(noise, col=alpha("#AOAOAO", 0.5),border = NA)
}

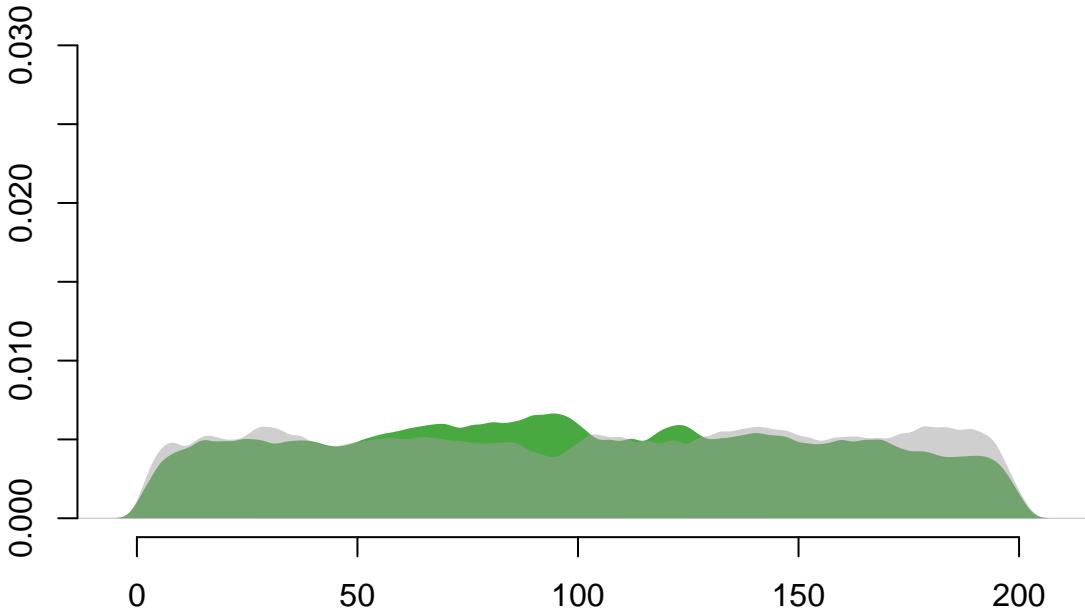
```











#### 0.5.4 Figure 2E

```

Names_protein <- c("HNRNPA1", "HNRNPC", "PTBP1", "RBF0X2", "YBX3")

#pdf("./Figure/Figure2E_similarity.pdf", width = 11, height = 11)
par(mfrow=c(5,3), mar=c(1,2,1,1))
set.seed(123)

for(i in Names_protein){
  # RAPSeq
  raps_rep1 <- log2(RAPs[[i]]$FCH_rep1)
  raps_rep2 <- log2(RAPs[[i]]$FCH_rep2)
  plot(raps_rep1, raps_rep2, xlim=c(0,10), ylim=c(0,10), xlab=NA, ylab=NA, bty="n", xaxt="n", yaxt="n",
    axis(1, at = c(0, 5, 10), lwd=1.5, labels = T)
  axis(2, at = c(0, 5, 10), lwd=1.5, labels = T)
  # spearman
  text_ann <- "rho ="
  raps_spearman <- round(cor(raps_rep1,raps_rep2, method = "spearman"), 2)
  text_ann <- paste(text_ann, raps_spearman, sep=" ")
  text(3,9.5,labels = raps_spearman, cex=1.5)

  # RAP sampling
  raps_sample <- RAPs[[i]][sample(1:nrow(RAPs[[i]]),(nrow(CLIPs[[i]]))),]
  raps_sample_rep1 <- log2(raps_sample$FCH_rep1)
}

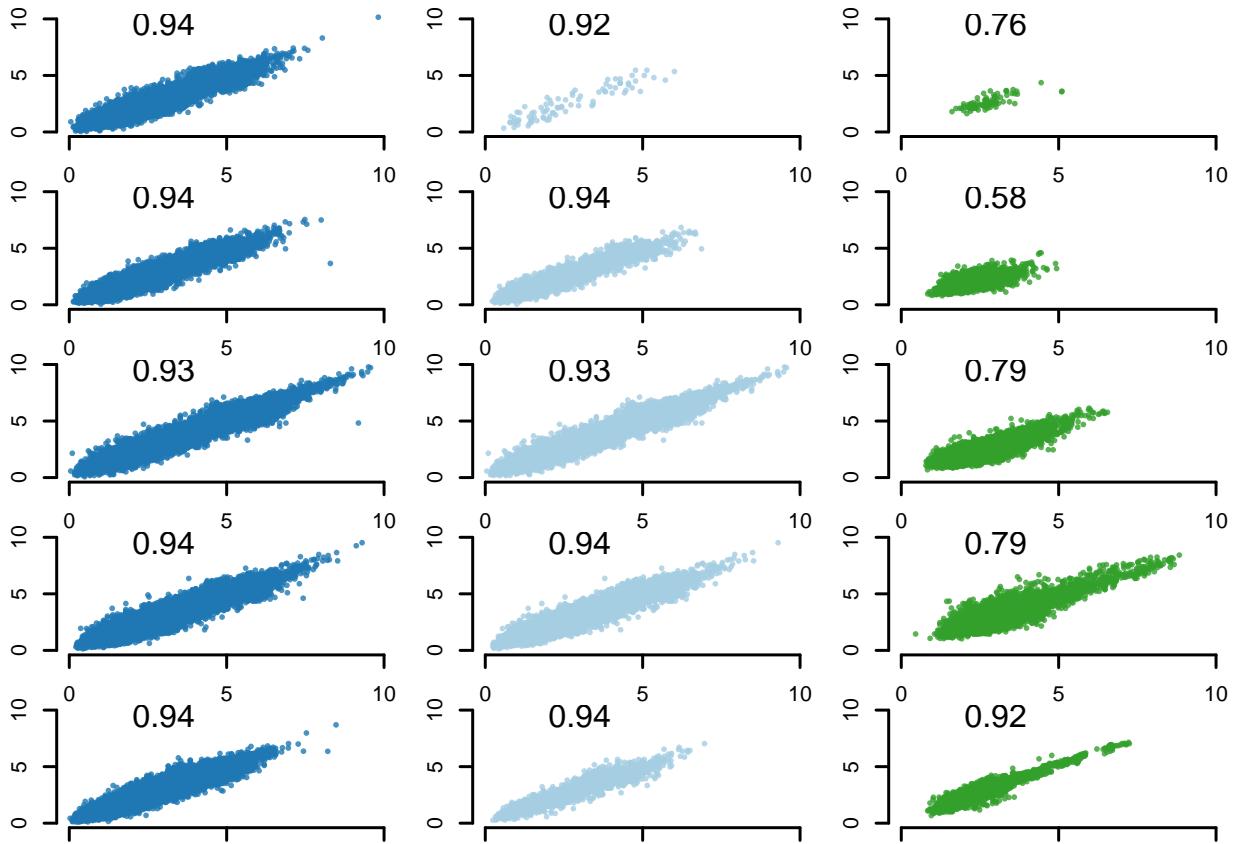
```

```

raps_sample_rep2 <- log2(raps_sample$FCH_rep2)
plot(raps_sample_rep1,raps_sample_rep2,xlim=c(0,10),ylim=c(0,10), xlab=NA, ylab=NA, bty="n", xaxt="n"
axis(1, at = c(0, 5, 10), lwd=1.5, labels = T)
axis(2, at = c(0, 5, 10), lwd=1.5, labels = T)
raps_sample_spearman <- round(cor(raps_sample_rep1,raps_sample_rep2, method = "spearman"), 2)
text_ann <- paste(text_ann, raps_sample_spearman, sep=" ")
text(3,9.5,labels = raps_sample_spearman, cex=1.5)

# eCLIP
eCLIP_rep1 <- log2(CLIPs[[i]]$FC_rep1)
eCLIP_rep2 <- log2(CLIPs[[i]]$FC_rep2)
plot(eCLIP_rep1, eCLIP_rep2, xlim=c(0,10),ylim=c(0,10), xlab=NA, ylab=NA, bty="n", xaxt="n", yaxt="n"
axis(1, at = c(0, 5, 10), lwd=1.5, labels = T)
axis(2, at = c(0, 5, 10), lwd=1.5, labels = T)
eCLIP_spearman <- round(cor(eCLIP_rep1,eCLIP_rep2, method = "spearman"), 2)
text_ann <- paste(text_ann, eCLIP_spearman,sep=" ")
text(3,9.5, labels = eCLIP_spearman, cex=1.5)
}

```



```
#dev.off()
```

### 0.5.5 Figure 2F

```

# spearman
RAP_spearman <- c()
CLIP_spearman <- c()
Names_protein <- c("HNRNPA1", "HNRNPC", "PTBP1", "RBFOX2", "YBX3")
for(i in Names_protein){
  raps_rep1 <- RAPs[[i]]$FCH_rep1
  raps_rep2 <- RAPs[[i]]$FCH_rep2
  RAP_spearman <- c(RAP_spearman, cor(raps_rep1, raps_rep2, method="spearman"))

  clip_rep1 <- CLIPs[[i]]$FC_rep1
  clip_rep2 <- CLIPs[[i]]$FC_rep2
  CLIP_spearman <- c(CLIP_spearman, cor(clip_rep1,clip_rep2,method="spearman"))
}

# merging
cors <- c(RAP_spearman,CLIP_spearman)
assay <- c(rep("aRAP",5),rep("eCLIP",5))

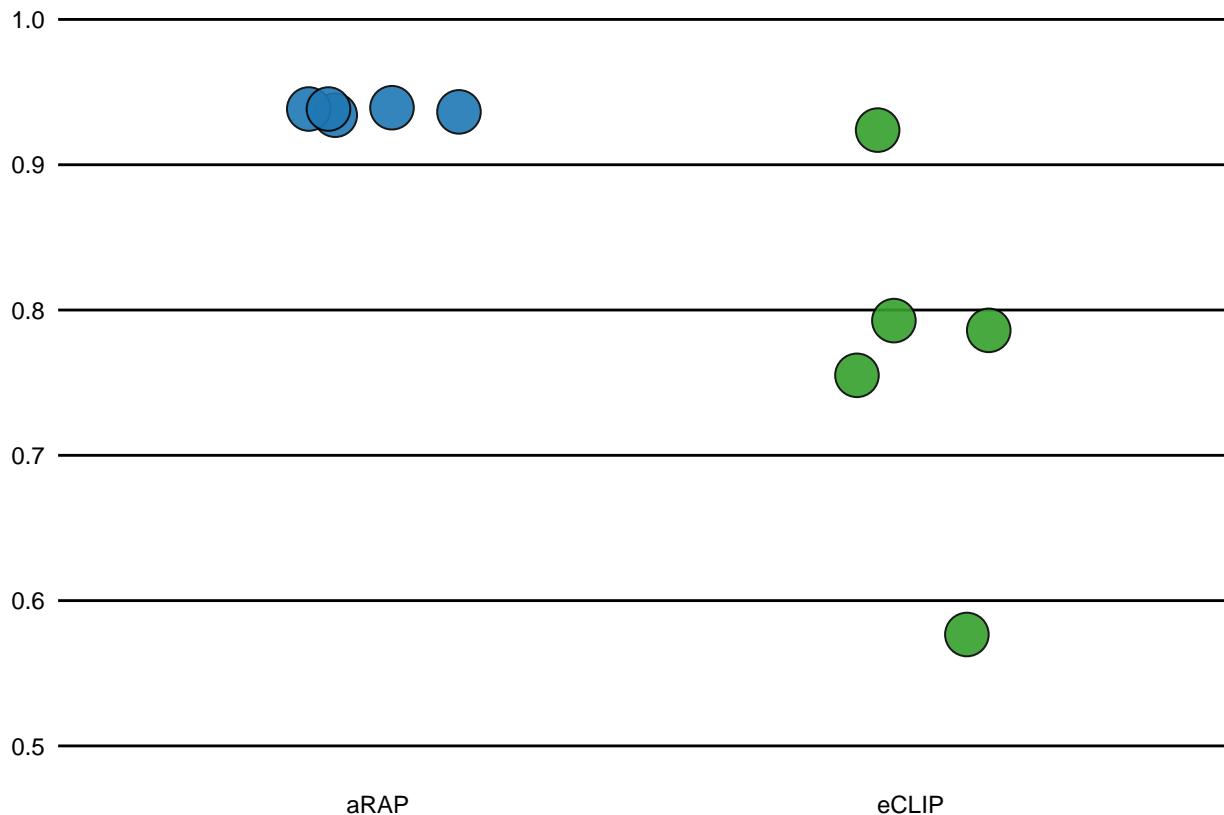
# data frame
data_cors <- data.frame(cors,assay)
data_cors$cors <- as.numeric(data_cors$cors)
data_cors$assay <- as.factor(data_cors$assay)

wilcox.test(data_cors[data_cors$assay == "aRAP",]$cors, data_cors[data_cors$assay == "eCLIP",]$cors)
##
## Wilcoxon rank sum exact test
##
## data: data_cors[data_cors$assay == "aRAP", ]$cors and data_cors[data_cors$assay == "eCLIP", ]$cors
## W = 25, p-value = 0.007937
## alternative hypothesis: true location shift is not equal to 0

# plot
set.seed(1)
#pdf("./Figure/Figure2_F.pdf", width = 2.9, height = 5.3)
ggplot(data=data_cors,aes(x=assay,y=cors,fill=assay)) +
  geom_dotplot(binaxis = "y",
               stackdir='center',
               binwidth = .005,
               dotsize = 6,
               position = position_jitter(height=NULL, width = 0.15),
               alpha = 0.9
  ) +
  theme(panel.background = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        panel.grid.major.y = element_line(color="black"),
        axis.ticks = element_blank(),
        axis.text = element_text(color="black"),
        legend.position = "none",
        axis.title = element_blank()
  ) +

```

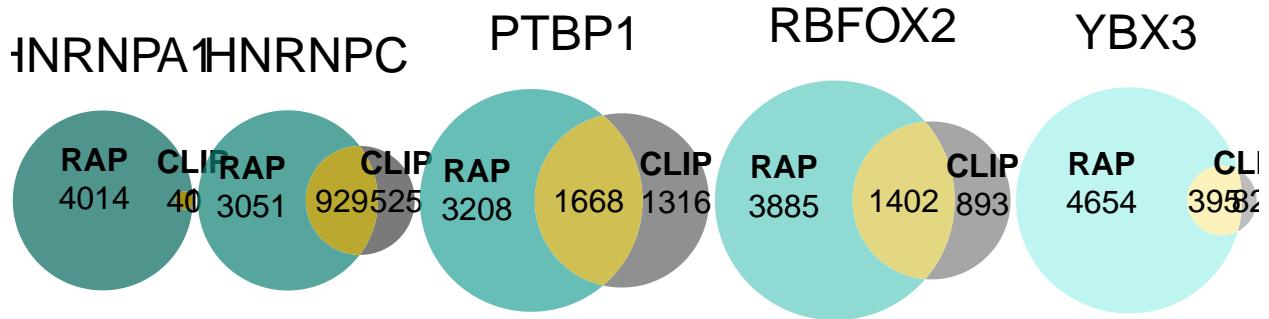
```
scale_fill_manual(values = c("#1f78b4", "#33a02c") ) +
ylim(0.5,1)
```



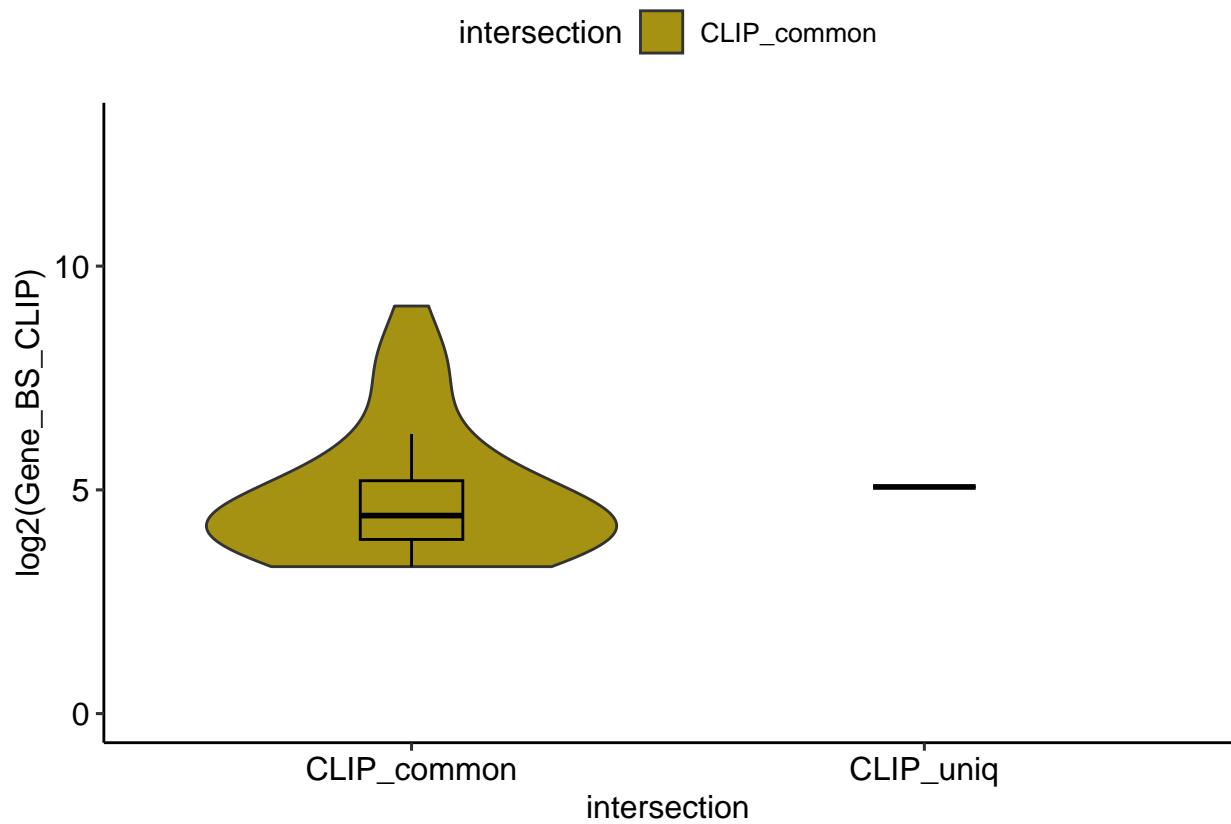
```
#dev.off()
```

### 0.5.6 Figure 2G

```
# left
venns <- list()
plot_area <- c()
Names_protein <- c("HNRNPA1", "HNRNPC", "PTBP1", "RBFOX2", "YBX3")
for (i in 1:length(Names_protein)){
  common_RAPs_CLIPs <- length(intersect(unique(RAPs[[Names_protein[i]]]$gene_ID), unique(CLIPs[[Names_protein[i]]]$gene_ID)))
  uniq_RAP <- length(unique(RAPs[[Names_protein[i]]]$gene_ID)) - common_RAPs_CLIPs
  uniq_CLIP <- length(unique(CLIPs[[Names_protein[i]]]$gene_ID)) - common_RAPs_CLIPs
  plot_area <- c(plot_area, length(unique(RAPs[[Names_protein[i]]]$gene_ID)) + common_RAPs_CLIPs)
  v <- euler(c(RAP=uniq_RAP, CLIP=uniq_CLIP, "RAP&CLIP"=common_RAPs_CLIPs))
  venns[[i]] <- plot(v, fills=c(alpha(acqua_greens[i*2],0.7),alpha(greys[i],0.7), yellows[i*2]), main="")
}
grid.arrange(grobs = venns, widths=plot_area)
```



```
# right
# HNRNPA1_pval
RAPS_HNRNPA1 <- unique(RAPS[["HNRNPA1"]][,c("gene_ID", "Gene_BS")])
CLIPS_HNRNPA1 <- unique(CLIPS[["HNRNPA1"]][,c("gene_ID", "Gene_BS_CLIP")])
RAPS_CLIPS_common_HNRNPA1 <- CLIPS_HNRNPA1[grep(paste(intersect(CLIPS_HNRNPA1$gene_ID, RAPS_HNRNPA1$gene_ID), collapse = ","), CLIPS_HNRNPA1$intersection) <- rep("CLIP_common", nrow(RAPS_CLIPS_common_HNRNPA1))]
CLIPS_uniq_HNRNPA1 <- CLIPS_HNRNPA1[grep(paste(setdiff(CLIPS_HNRNPA1$gene_ID, RAPS_HNRNPA1$gene_ID), collapse = ","), CLIPS_HNRNPA1$intersection) <- rep("CLIP_uniq", nrow(CLIPS_uniq_HNRNPA1))]
CLIPS_common_uniq_HNRNPA1 <- rbind(RAPS_CLIPS_common_HNRNPA1, CLIPS_uniq_HNRNPA1)
plot_HNRNPA1 <- ggplot( data = CLIPS_common_uniq_HNRNPA1, aes( x=intersection,y=log2(Gene_BS_CLIP) ) ) +
  geom_violin(aes(fill = intersection), bw=0.85, width=0.8) +
  scale_fill_manual(values = c(yellows[2],greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_HNRNPA1
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

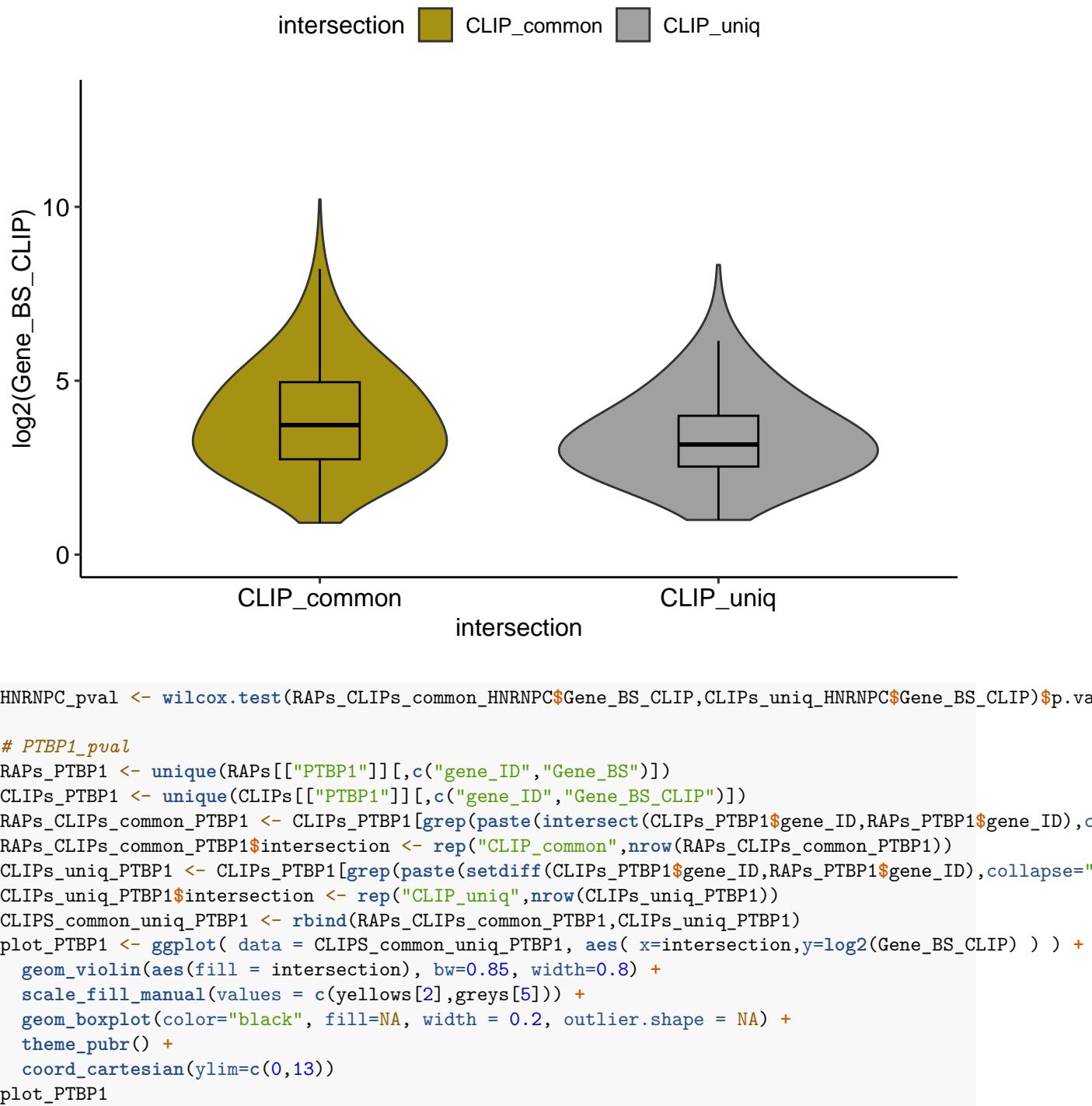


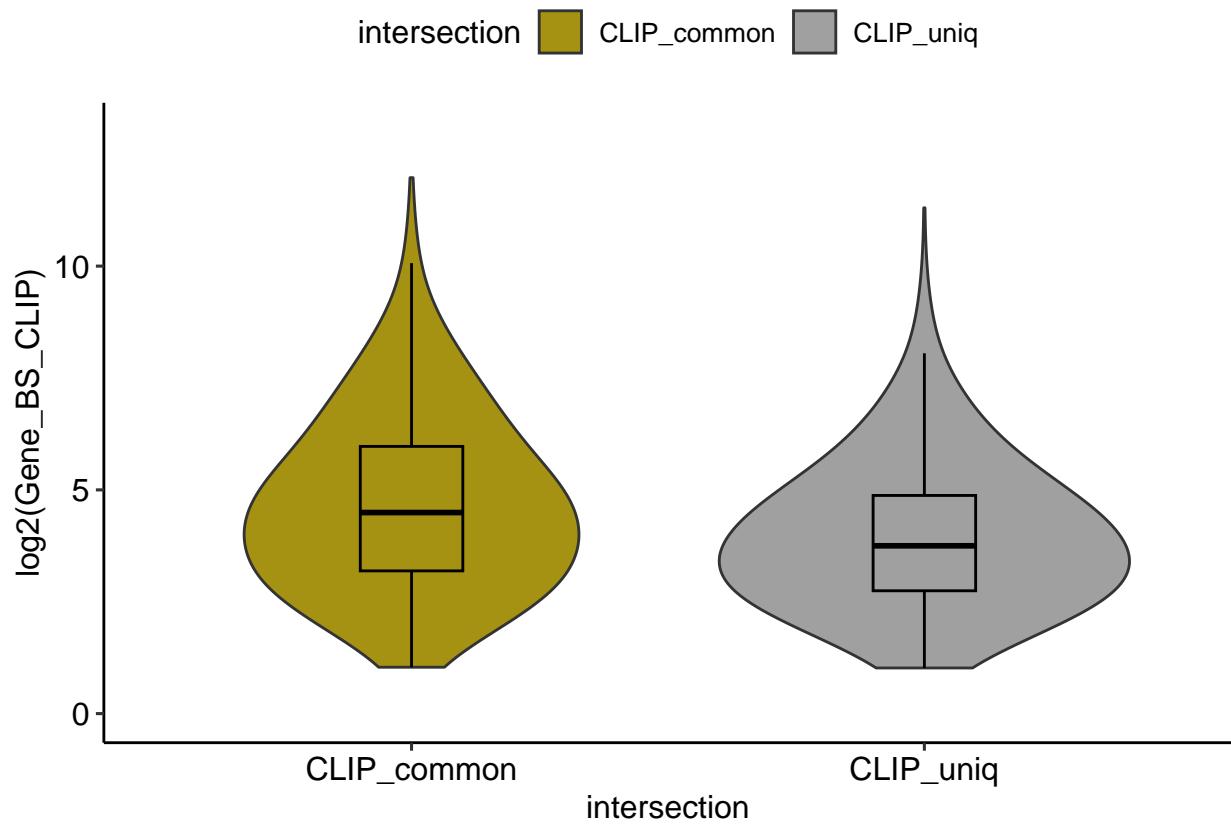
```

HNRNPA1_pval <- wilcox.test(RAPs_CLIPs_common_HNRNPA1$Gene_BS_CLIP, CLIPs_uniq_HNRNPA1$Gene_BS_CLIP)$p.value

# HNRNPC_pval
RAPs_HNRNPC <- unique(RAPs[["HNRNPC"]][,c("gene_ID", "Gene_BS")])
CLIPs_HNRNPC <- unique(CLIPs[["HNRNPC"]][,c("gene_ID", "Gene_BS_CLIP")])
RAPs_CLIPs_common_HNRNPC <- CLIPs_HNRNPC[grep(paste(intersect(CLIPs_HNRNPC$gene_ID, RAPs_HNRNPC$gene_ID), "CLIP_common"), nrow(RAPs_CLIPs_common_HNRNPC))]
CLIPs_uniq_HNRNPC <- CLIPs_HNRNPC[grep(paste(setdiff(CLIPs_HNRNPC$gene_ID, RAPs_HNRNPC$gene_ID), collapse="CLIP_uniq"), nrow(CLIPs_uniq_HNRNPC))]
CLIPs_common_uniq_HNRNPC <- rbind(RAPs_CLIPs_common_HNRNPC, CLIPs_uniq_HNRNPC)
plot_HNRNPC <- ggplot( data = CLIPs_common_uniq_HNRNPC, aes( x=intersection, y=log2(Gene_BS_CLIP) ) ) +
  geom_violin(aes(fill = intersection), bw=0.85, width=0.8) +
  scale_fill_manual(values = c(yellows[2], greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_HNRNPC

```



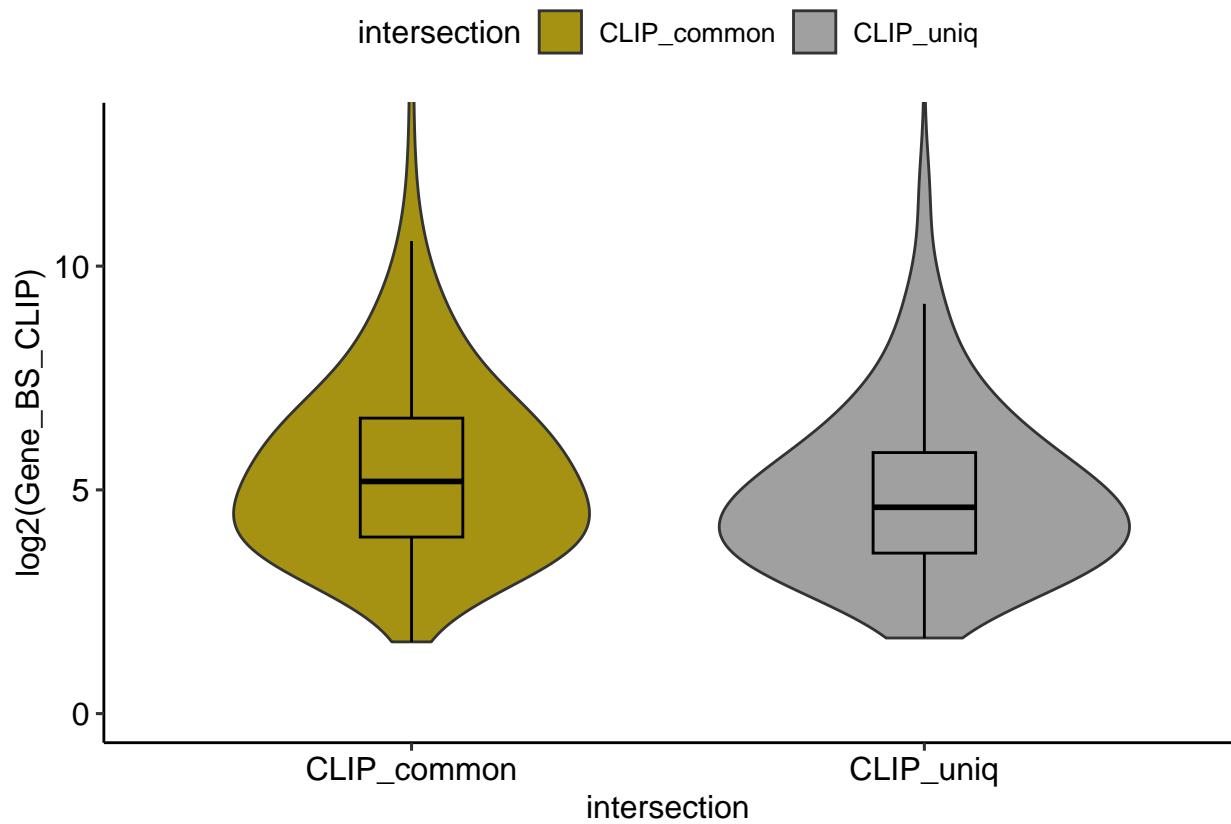


```

PTBP1_pval <- wilcox.test(RAPs_CLIPs_common_PTBP1$Gene_BS_CLIP,CLIPs_uniq_PTBP1$Gene_BS_CLIP)$p.value

# RBFOX2_pval
RAPs_RBFOX2 <- unique(RAPs[["RBFOX2"]][,c("gene_ID","Gene_BS")])
CLIPs_RBFOX2 <- unique(CLIPs[["RBFOX2"]][,c("gene_ID","Gene_BS_CLIP")])
RAPs_CLIPs_common_RBFOX2 <- CLIPs_RBFOX2[grep(paste(intersect(CLIPs_RBFOX2$gene_ID,RAPs_RBFOX2$gene_ID),
RAPs_CLIPs_common_RBFOX2$intersection <- rep("CLIP_common",nrow(RAPs_CLIPs_common_RBFOX2))
CLIPs_uniq_RBFOX2 <- CLIPs_RBFOX2[grep(paste(setdiff(CLIPs_RBFOX2$gene_ID,RAPs_RBFOX2$gene_ID),collapse=""),
CLIPs_uniq_RBFOX2$intersection <- rep("CLIP_uniq",nrow(CLIPs_uniq_RBFOX2))
CLIPs_common_uniq_RBFOX2 <- rbind(RAPs_CLIPs_common_RBFOX2,CLIPs_uniq_RBFOX2)
plot_RBFOX2 <- ggplot( data = CLIPs_common_uniq_RBFOX2, aes( x=intersection,y=log2(Gene_BS_CLIP) ) ) +
  geom_violin(aes(fill = intersection), bw=0.85, width=0.8) +
  scale_fill_manual(values = c(yellows[2],greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_RBFOX2

```

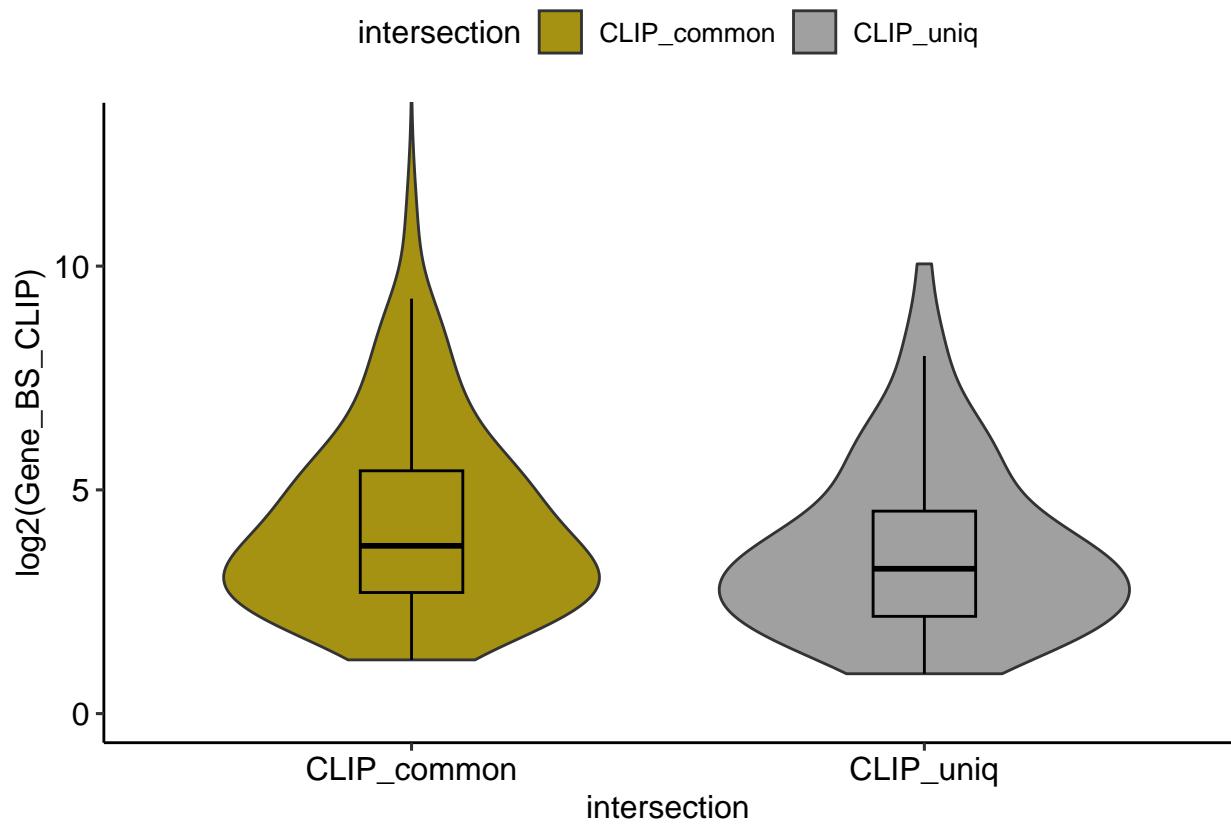


```

RBFOX2_pval <- wilcox.test(RAPs_CLIPs_common_RBFOX2$Gene_BS_CLIP, CLIPs_uniq_RBFOX2$Gene_BS_CLIP)$p.value

# YBX3_pval
RAPs_YBX3 <- unique(RAPs[["YBX3"]][,c("gene_ID", "Gene_BS")])
CLIPs_YBX3 <- unique(CLIPs[["YBX3"]][,c("gene_ID", "Gene_BS_CLIP")])
RAPs_CLIPs_common_YBX3 <- CLIPs_YBX3[grep(paste(intersect(CLIPs_YBX3$gene_ID, RAPs_YBX3$gene_ID), collapse="|"), CLIPs_YBX3$intersection) <- rep("CLIP_common", nrow(RAPs_CLIPs_common_YBX3))
CLIPs_uniq_YBX3 <- CLIPs_YBX3[grep(paste(setdiff(CLIPs_YBX3$gene_ID, RAPs_YBX3$gene_ID), collapse="|"), CLIPs_YBX3$intersection) <- rep("CLIP_uniq", nrow(CLIPs_uniq_YBX3))
CLIPs_common_uniq_YBX3 <- rbind(RAPs_CLIPs_common_YBX3, CLIPs_uniq_YBX3)
plot_YBX3 <- ggplot( data = CLIPs_common_uniq_YBX3, aes( x=intersection,y=log2(Gene_BS_CLIP) ) ) +
  geom_violin(aes(fill = intersection), bw=0.85, width=0.8) +
  scale_fill_manual(values = c(yellows[2],greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_YBX3

```



```

YBX3_pval <- wilcox.test(RAPs_CLIPs_common_YBX3$Gene_BS_CLIP,CLIPs_uniq_YBX3$Gene_BS_CLIP)$p.value

# all
print("Wilcoxon rank sum test p values")
## [1] "Wilcoxon rank sum test p values"
print(paste("HNRNPA1",HNRNPA1_pval, sep=" "))
## [1] "HNRNPA1 0.585365853658537"
print(paste("HNRNPC",HNRNPC_pval,sep=" "))
## [1] "HNRNPC 5.79007340906723e-11"
print(paste("PTBP1",PTBP1_pval,sep=" "))
## [1] "PTBP1 1.1011492584217e-25"
print(paste("RBFOX2",RBFOX2_pval,sep=" "))
## [1] "RBFOX2 1.3676840269862e-12"
print(paste("YBX3",YBX3_pval,sep=" "))
## [1] "YBX3 0.00451598791860509"

p_figure_violin_2G <- ggarrange(plot_HNRNPA1,plot_HNRNPC,plot_PTBP1,plot_RBFOX2,plot_YBX3,nrow=5, common
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
## Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
ggsave(filename = "./Figure/Figure2G_Violin.pdf", plot = p_figure_violin_2G, width = 4.7, height = 12.6)

CLIPS_common_uniq_HNRNPA1 <- CLIPS_common_uniq_HNRNPA1[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
CLIPS_common_uniq_HNRNPC <- CLIPS_common_uniq_HNRNPC[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
CLIPS_common_uniq_PTBP1 <- CLIPS_common_uniq_PTBP1[,c("gene_ID", "intersection", "Gene_BS_CLIP")]

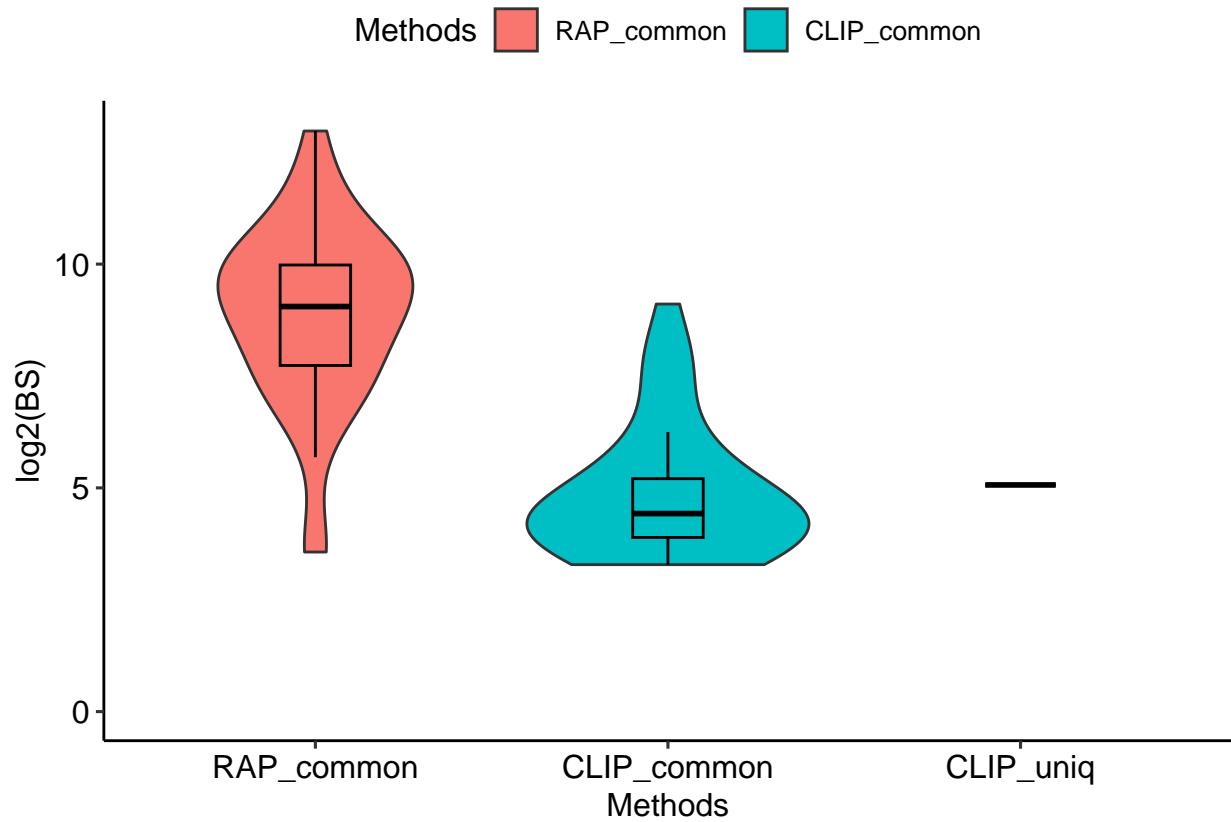
```

```

CLIPS_common_uniq_RBFOX2 <- CLIPS_common_uniq_RBFOX2[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
CLIPS_common_uniq_YBX3 <- CLIPS_common_uniq_YBX3[,c("gene_ID", "intersection", "Gene_BS_CLIP")]

# HNRNPA1
HNRNPA1_CLIPS_RAPS_Qun <- merge(RAPs_HNRNPA1, CLIPs_HNRNPA1, by = "gene_ID")
colnames(HNRNPA1_CLIPS_RAPS_Qun) <- c("gene_ID", "RAP_common", "Gene_BS_CLIP")
HNRNPA1_CLIPS_RAPS_Qun_long <- pivot_longer(
  HNRNPA1_CLIPS_RAPS_Qun,
  cols = -gene_ID,
  names_to = "Methods",
  values_to = "BS"
)
#CLIPS_common_uniq_HNRNPA1 <- CLIPS_common_uniq_HNRNPA1[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
colnames(CLIPS_common_uniq_HNRNPA1) <- c("gene_ID", "Methods", "BS")
data_HNRNPA1_comparision_Qun <- rbind(HNRNPA1_CLIPS_RAPS_Qun_long, CLIPS_common_uniq_HNRNPA1)
data_HNRNPA1_comparision_Qun <- data_HNRNPA1_comparision_Qun[data_HNRNPA1_comparision_Qun$Methods != "G"]
data_HNRNPA1_comparision_Qun$Methods <- factor(data_HNRNPA1_comparision_Qun$Methods, levels = c("RAP_common", "BS"))
plot_HNRNPA1 <- ggplot( data = data_HNRNPA1_comparision_Qun, aes( x=Methods,y=log2(BS) ) ) +
  geom_violin(aes(fill = Methods), bw=0.85, width=0.8) +
  #scale_fill_manual(values = c(yellows[2],greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_HNRNPA1
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.

```

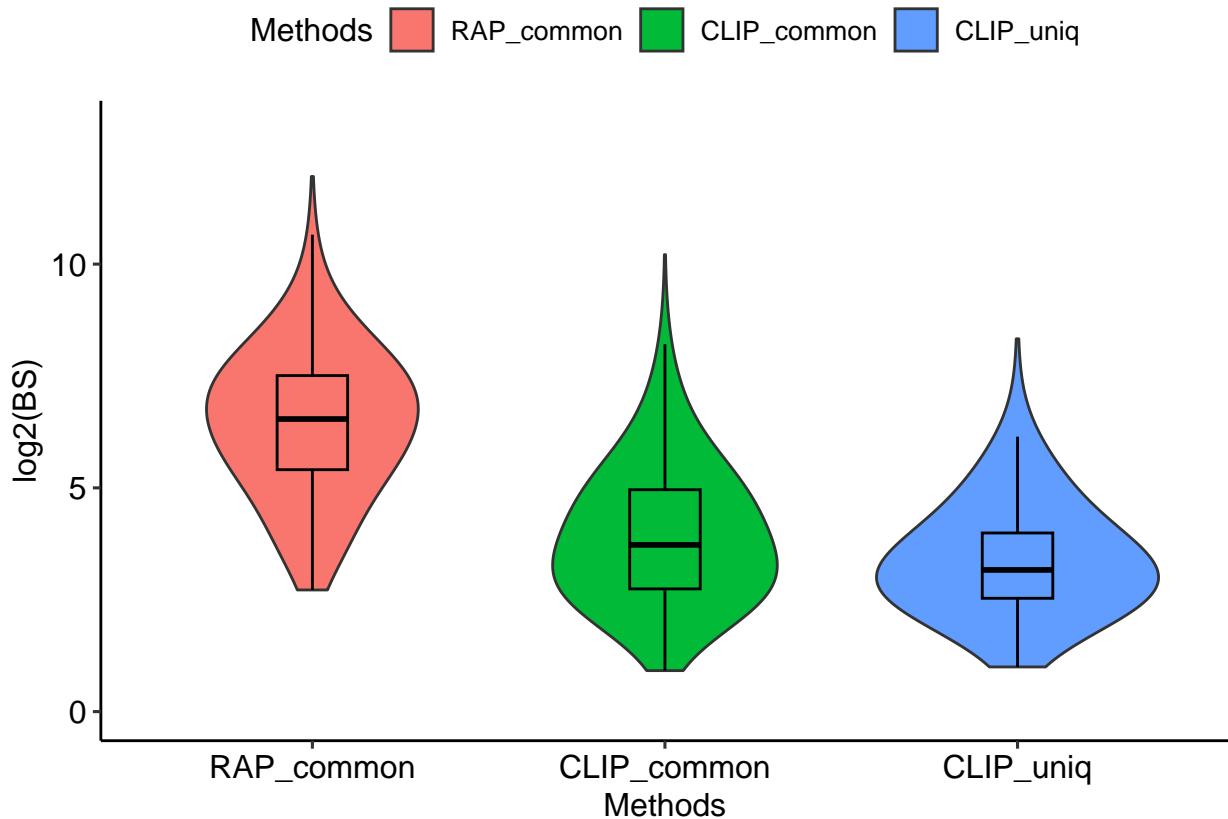


```

HNRNPA1_pval <- wilcox.test(HNRNPA1_CLIPS_RAPS_Qun$RAP_common,HNRNPA1_CLIPS_RAPS_Qun$Gene_BS_CLIP)$p.value

# HNRNPC
HNRNPC_CLIPS_RAPS_Qun <- merge(RAPs_HNRNPC, CLIPs_HNRNPC, by = "gene_ID")
colnames(HNRNPC_CLIPS_RAPS_Qun) <- c("gene_ID", "RAP_common", "Gene_BS_CLIP")
HNRNPC_CLIPS_RAPS_Qun_long <- pivot_longer(
  HNRNPC_CLIPS_RAPS_Qun,
  cols = -gene_ID,
  names_to = "Methods",
  values_to = "BS"
)
#CLIPS_common_uniq_HNRNPC <- CLIPS_common_uniq_HNRNPC[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
colnames(CLIPS_common_uniq_HNRNPC) <- c("gene_ID", "Methods", "BS")
data_HNRNPC_comparision_Qun <- rbind(HNRNPC_CLIPS_RAPS_Qun_long, CLIPS_common_uniq_HNRNPC)
data_HNRNPC_comparision_Qun <- data_HNRNPC_comparision_Qun[data_HNRNPC_comparision_Qun$Methods != "Gene_BS_CLIP"]
data_HNRNPC_comparision_Qun$Methods <- factor(data_HNRNPC_comparision_Qun$Methods, levels = c("RAP_common", "CLIP_common"))
plot_HNRNPC <- ggplot( data = data_HNRNPC_comparision_Qun, aes( x=Methods,y=log2(BS) ) ) +
  geom_violin(aes(fill = Methods), bw=0.85, width=0.8) +
  #scale_fill_manual(values = c(yellows[2],greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_HNRNPC

```

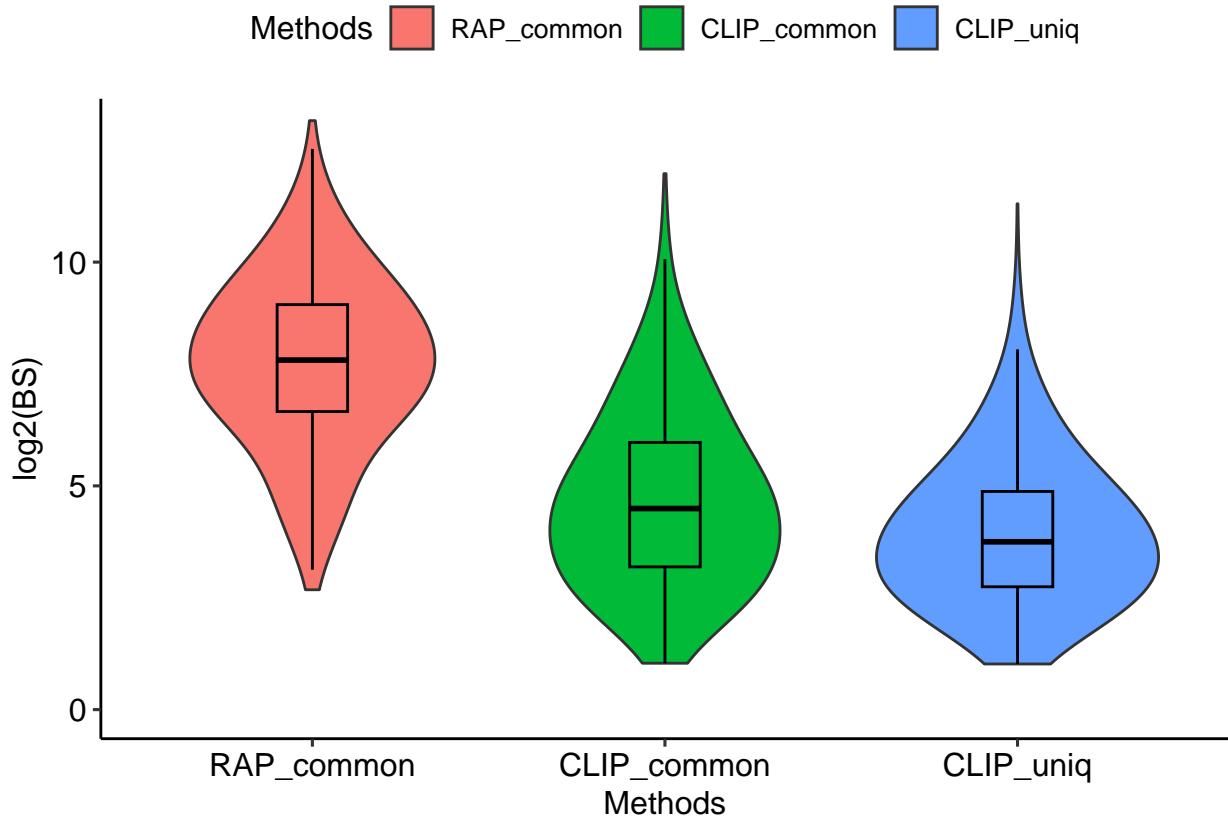


```

HNRNPC_pval <- wilcox.test(HNRNPC_CLIPS_RAPS_Qun$RAP_common,HNRNPC_CLIPS_RAPS_Qun$Gene_BS_CLIP)$p.value

# PTBP1
PTBP1_CLIPS_RAPS_Qun <- merge(RAPs_PTBP1, CLIPs_PTBP1, by = "gene_ID")
colnames(PTBP1_CLIPS_RAPS_Qun) <- c("gene_ID", "RAP_common", "Gene_BS_CLIP")
PTBP1_CLIPS_RAPS_Qun_long <- pivot_longer(
  PTBP1_CLIPS_RAPS_Qun,
  cols = -gene_ID,
  names_to = "Methods",
  values_to = "BS"
)
#CLIPS_common_uniq_PTBP1 <- CLIPS_common_uniq_PTBP1[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
colnames(CLIPS_common_uniq_PTBP1) <- c("gene_ID", "Methods", "BS")
data_PTBP1_comparision_Qun <- rbind(PTBP1_CLIPS_RAPS_Qun_long, CLIPS_common_uniq_PTBP1)
data_PTBP1_comparision_Qun <- data_PTBP1_comparision_Qun[data_PTBP1_comparision_Qun$Methods != "Gene_BS"]
data_PTBP1_comparision_Qun$Methods <- factor(data_PTBP1_comparision_Qun$Methods, levels = c("RAP_common", "CLIP_common", "CLIP_uniq"))
plot_PTBP1 <- ggplot( data = data_PTBP1_comparision_Qun, aes( x=Methods,y=log2(BS) ) ) +
  geom_violin(aes(fill = Methods), bw=0.85, width=0.8) +
  #scale_fill_manual(values = c(yellows[2],greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_PTBP1

```

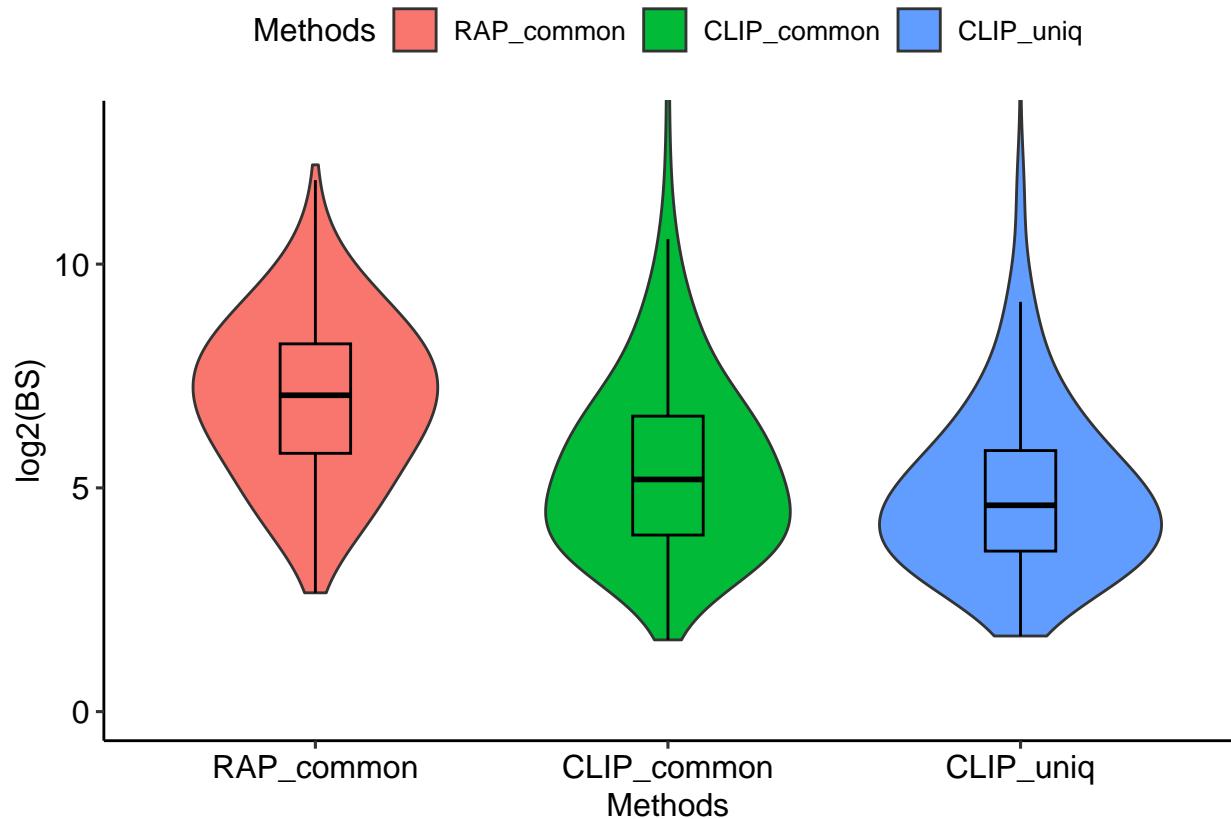


```

PTBP1_pval <- wilcox.test(PTBP1_CLIPS_RAPS_Qun$RAP_common, PTBP1_CLIPS_RAPS_Qun$Gene_BS_CLIP)$p.value

# RBFOX2
RBFOX2_CLIPS_RAPS_Qun <- merge(RAPs_RBFOX2, CLIPs_RBFOX2, by = "gene_ID")
colnames(RBFOX2_CLIPS_RAPS_Qun) <- c("gene_ID", "RAP_common", "Gene_BS_CLIP")
RBFOX2_CLIPS_RAPS_Qun_long <- pivot_longer(
  RBFOX2_CLIPS_RAPS_Qun,
  cols = -gene_ID,
  names_to = "Methods",
  values_to = "BS"
)
#CLIPS_common_uniq_RBFOX2 <- CLIPS_common_uniq_RBFOX2[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
colnames(CLIPS_common_uniq_RBFOX2) <- c("gene_ID", "Methods", "BS")
data_RBFOX2_comparision_Qun <- rbind(RBFOX2_CLIPS_RAPS_Qun_long, CLIPS_common_uniq_RBFOX2)
data_RBFOX2_comparision_Qun <- data_RBFOX2_comparision_Qun[data_RBFOX2_comparision_Qun$Methods != "Gene"]
data_RBFOX2_comparision_Qun$Methods <- factor(data_RBFOX2_comparision_Qun$Methods, levels = c("RAP_common", "CLIP_common", "CLIP_uniq"))
plot_RBFOX2 <- ggplot( data = data_RBFOX2_comparision_Qun, aes( x=Methods,y=log2(BS) ) ) +
  geom_violin(aes(fill = Methods), bw=0.85, width=0.8) +
  #scale_fill_manual(values = c(yellows[2], greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_RBFOX2

```

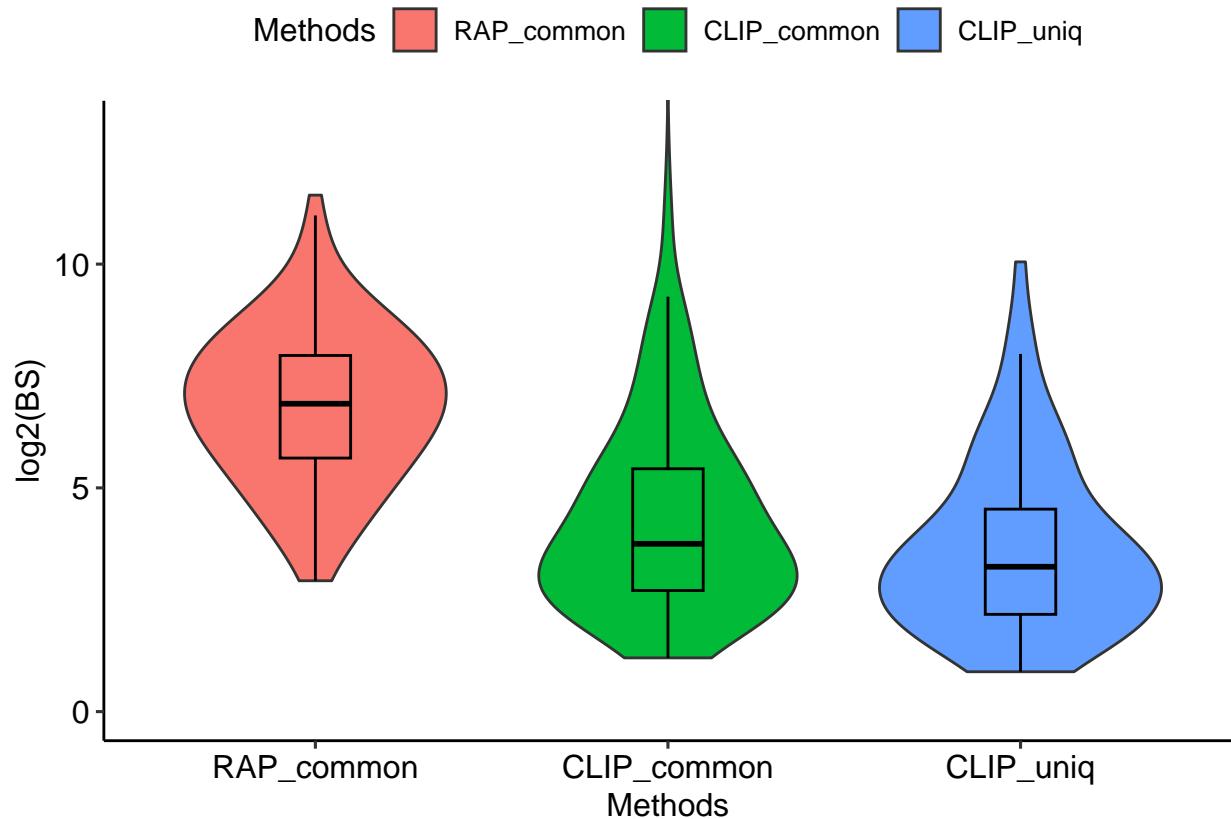


```

RBFOX2_pval <- wilcox.test(RBFOX2_CLIPS_RAPS_Qun$RAP_common,RBFOX2_CLIPS_RAPS_Qun$Gene_BS_CLIP)$p.value

# YBX3
YBX3_CLIPS_RAPS_Qun <- merge(RAPs_YBX3, CLIPs_YBX3, by = "gene_ID")
colnames(YBX3_CLIPS_RAPS_Qun) <- c("gene_ID", "RAP_common", "Gene_BS_CLIP")
YBX3_CLIPS_RAPS_Qun_long <- pivot_longer(
  YBX3_CLIPS_RAPS_Qun,
  cols = -gene_ID,
  names_to = "Methods",
  values_to = "BS"
)
#CLIPS_common_uniq_YBX3 <- CLIPS_common_uniq_YBX3[,c("gene_ID", "intersection", "Gene_BS_CLIP")]
colnames(CLIPS_common_uniq_YBX3) <- c("gene_ID", "Methods", "BS")
data_YBX3_comparision_Qun <- rbind(YBX3_CLIPS_RAPS_Qun_long, CLIPS_common_uniq_YBX3)
data_YBX3_comparision_Qun <- data_YBX3_comparision_Qun[data_YBX3_comparision_Qun$Methods != "Gene_BS_CLIP"]
data_YBX3_comparision_Qun$Methods <- factor(data_YBX3_comparision_Qun$Methods, levels = c("RAP_common",
plot_YBX3 <- ggplot( data = data_YBX3_comparision_Qun, aes( x=Methods,y=log2(BS) ) ) +
  geom_violin(aes(fill = Methods), bw=0.85, width=0.8) +
  #scale_fill_manual(values = c(yellows[2],greys[5])) +
  geom_boxplot(color="black", fill=NA, width = 0.2, outlier.shape = NA) +
  theme_pubr() +
  coord_cartesian(ylim=c(0,13))
plot_YBX3

```



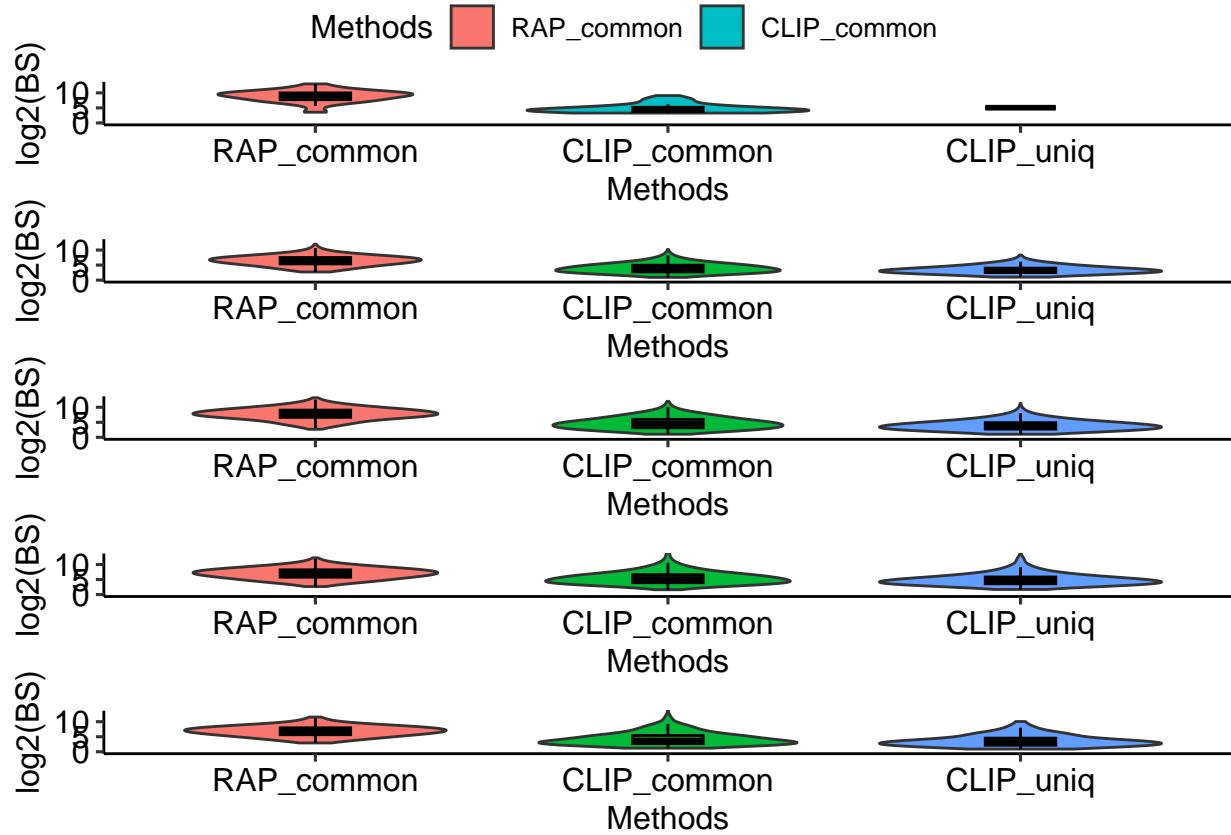
```

YBX3_pval <- wilcox.test(YBX3_CLIPS_RAPS_Qun$RAP_common,YBX3_CLIPS_RAPS_Qun$Gene_BS_CLIP)$p.value

print("Wilcoxon rank sum test p values")
## [1] "Wilcoxon rank sum test p values"
print(paste("HNRNPA1",HNRNPA1_pval, sep=" "))
## [1] "HNRNPA1 4.04267121005842e-13"
print(paste("HNRNPC",HNRNPC_pval,sep=" "))
## [1] "HNRNPC 4.22620979968423e-171"
print(paste("PTBP1",PTBP1_pval,sep=" "))
## [1] "PTBP1 6.42758024429207e-294"
print(paste("RBFOX2",RBFOX2_pval,sep=" "))
## [1] "RBFOX2 1.79421281391685e-98"
print(paste("YBX3",YBX3_pval,sep=" "))
## [1] "YBX3 5.32399130122868e-58"

p_figure_violin_2G <- ggarrange(plot_HNRNPA1,plot_HNRNPC,plot_PTBP1,plot_RBFOX2,plot_YBX3,nrow=5, common
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
## Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
p_figure_violin_2G

```



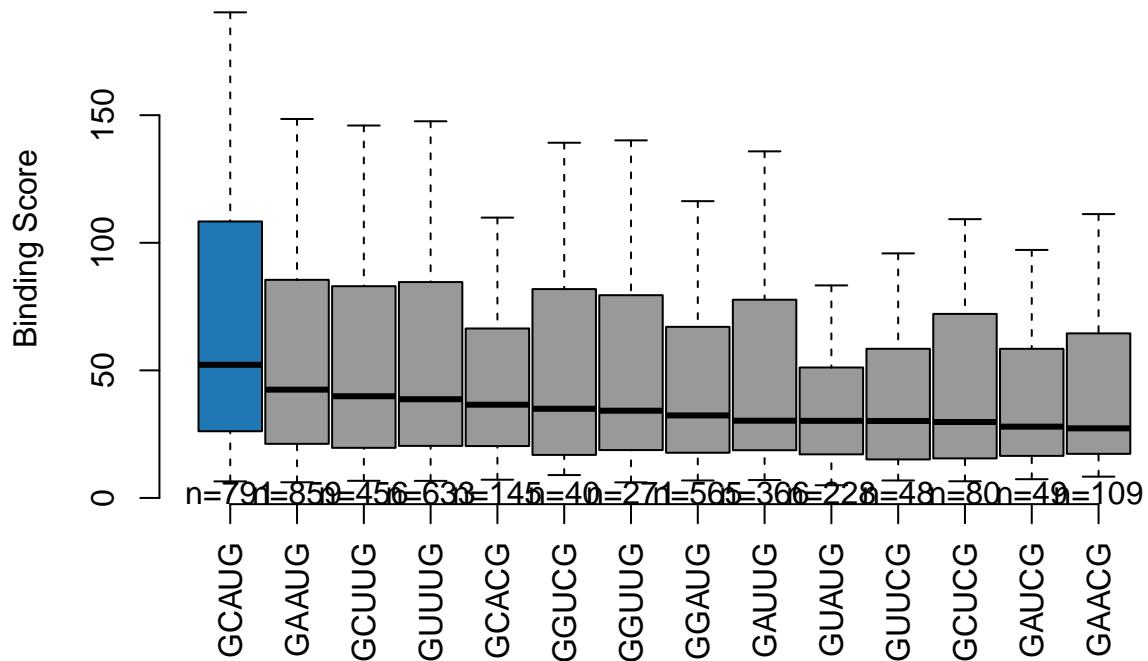
```
ggsave(filename = "./Figure/Figure2G_Violin_Qun.pdf", plot = p_figure_violin_2G, width = 4.7, height = 1)
```

### 0.5.7 Figure S2B

```
GNWYG <- c("GCATG", "GAATG", "GGATG", "GTTTG", "GCTTG", "GATTG", "GGTTG", "GAACG", "GCACG", "GTATG", "GATCG", "GCT")
GNWYG_scores <- list()
medians <- c()
for (i in 1:length(GNWYG)){
  RAPs_GNWYG <- RAPs[["RBFOX2"]][grep(GNWYG[i], str_sub(RAPs[["RBFOX2"]]$positive_fa, 50, 150)),]
  GNWYG_subset <- GNWYG[grep(GNWYG[i], GNWYG, invert=T)]
  GNWYG_scores[[i]] <- RAPs_GNWYG[grep(paste(GNWYG_subset, collapse = "|"), str_sub(RAPs_GNWYG$positive_fa, 50, 150))]]
  medians <- c(medians, median(GNWYG_scores[[i]])))
}

GNWYG <- gsub("T", "U", GNWYG)
names(GNWYG_scores) <- GNWYG

#pdf("./Figure/Figure2/FigureS2B.pdf", width = 11, height = 7)
par(bty="n")
boxplot2(GNWYG_scores[order(-medians)], outline=F, horizontal=F, las=3, ylab="Binding Score", range=1, cex.lab=1.5, cex.axis=1.5, cex=1.5)
```



```
#dev.off()

GCAUG <- GNWYG_scores$GCAUG
GNWYG_scores <- GNWYG_scores[-1]

for (i in names(GNWYG_scores)){
  anno_text_0 <- paste("GCAUG vs", i)
  test_compare <- wilcox.test(GCAUG, GNWYG_scores[[i]])$p.value
  anno_text <- paste("P-value =", test_compare)
  anno_text <- paste(anno_text_0, anno_text, sep=" ")
  print(anno_text)
}
## [1] "GCAUG vs GAAUG P-value = 0.000187642717483513"
## [1] "GCAUG vs GGAUG P-value = 1.75027346808954e-14"
## [1] "GCAUG vs GCUUG P-value = 1.34540298133905e-05"
## [1] "GCAUG vs GUUUG P-value = 4.53503535262265e-05"
## [1] "GCAUG vs GACCG P-value = 3.61588355304033e-09"
## [1] "GCAUG vs GGUUG P-value = 4.84423104579228e-06"
## [1] "GCAUG vs GAAACG P-value = 2.25248427316068e-06"
## [1] "GCAUG vs GCACG P-value = 0.000236877780284992"
## [1] "GCAUG vs GUAUG P-value = 3.30289723362946e-13"
## [1] "GCAUG vs GAUCG P-value = 0.00021981847552523"
## [1] "GCAUG vs GCUCG P-value = 4.55398275989635e-05"
## [1] "GCAUG vs GUUCG P-value = 0.000457210969649777"
## [1] "GCAUG vs GGUCG P-value = 0.0813437382604083"
```

### 0.5.8 Figure S2C

```

GNWYG <- c("GCATG", "GAATG", "GGATG", "GTTTG", "GCTTG", "GATTG", "GGTTG", "GAACG", "GCACG", "GTATG", "GATCG", "GCT
GNWYG <- GNWYG[order(-medians)]

#pdf("./Figure/Figure2/FigureS2C.pdf", width = 10, height = 8)
par(mfrow=c(3,5), mar=c(2,3.5,2,0.5))

for (i in GNWYG){
  ymax <- round(nrow(RAPs[["RBFOX2"]][grep(i,str_sub(RAPs[["RBFOX2"]]$positive_fa,50,150)),]) / 2)

  hist(unlist(str_locate_all(RAPs[["RBFOX2"]]$positive_fa,i))-100, breaks=40, main=gsub("T","U",i), yla
axis(side = 1, at = c(-100,-50,0,50,100), labels = c("-100",NA,"0",NA,"100"))
x <- nrow(RAPs[["RBFOX2"]][grep(i,str_sub(RAPs[["RBFOX2"]]$positive_fa,80,120)),])
n <- nrow(RAPs[["RBFOX2"]][grep(i,RAPs[["RBFOX2"]]$positive_fa),])

  hist(unlist(str_locate_all(RAPs[["HNRNPA1"]]$positive_fa,i))-100, breaks=40, type="l",col=alpha("#ff7f
n_neg <- nrow(RAPs[["HNRNPA1"]][grep(i,RAPs[["HNRNPA1"]]$positive_fa),])
x_neg <- nrow(RAPs[["HNRNPA1"]][grep(i,str_sub(RAPs[["HNRNPA1"]]$positive_fa,80,120)),])
p <- x_neg / n_neg
bt <- binom.test(x,n,p, alternative = "greater")
btp <- bt$p.value
if ( btp <= 0.001 ) {
  to_add <- "***"
} else if ( btp <= 0.01 ) {
  to_add <- "**"
} else if ( btp <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "n.s."
}
text(50,ymax/10*9,labels = to_add, cex=2.5, col = yellows[1])

  hist(unlist(str_locate_all(RAPs[["HNRNPC"]]$positive_fa,i))-100, breaks=40, type="l",col=alpha("#6a3d
n_neg <- nrow(RAPs[["HNRNPC"]][grep(i,RAPs[["HNRNPC"]]$positive_fa),])
x_neg <- nrow(RAPs[["HNRNPC"]][grep(i,str_sub(RAPs[["HNRNPC"]]$positive_fa,80,120)),])
p <- x_neg / n_neg
bt <- binom.test(x,n,p, alternative = "greater")
btp <- bt$p.value
if ( btp <= 0.001 ) {
  to_add <- "***"
} else if ( btp <= 0.01 ) {
  to_add <- "**"
} else if ( btp <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "n.s."
}
text(50,ymax/10*7.75,labels = to_add, cex=2.5, col = yellows[2])

```

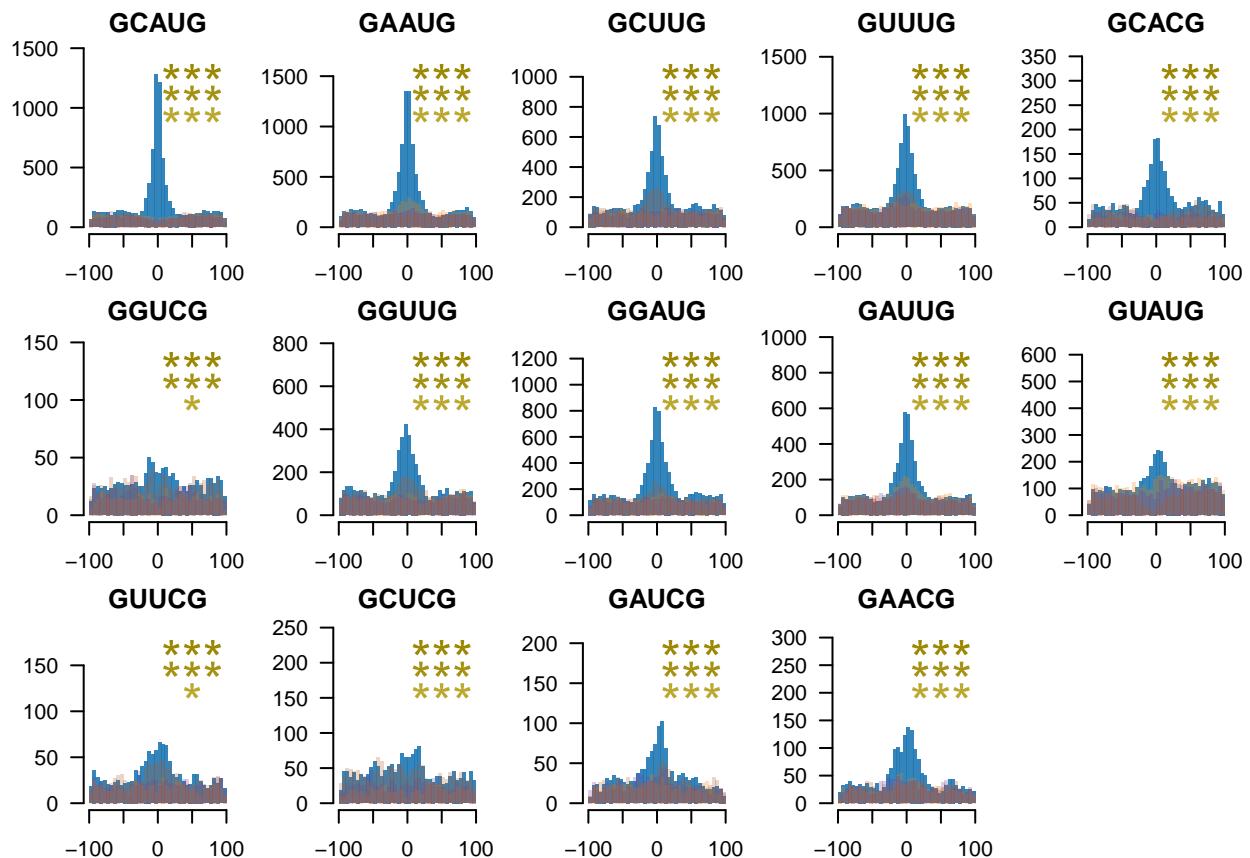
```

hist(unlist(str_locate_all(RAPs[["PTBP1"]]$positive_fa,i))-100, breaks=40, type="l", col=alpha("#b1592d"))
n_neg <- nrow(RAPs[["PTBP1"]][grep(i,RAPs[["PTBP1"]]$positive_fa),])
x_neg <- nrow(RAPs[["PTBP1"]][grep(i,str_sub(RAPs[["PTBP1"]]$positive_fa,80,120)),])
p <- x_neg / n_neg
#bt <- binom.test(x,n,p, alternative = "greater")
bt <- binom.test(x,n,p)
btp <- bt$p.value
if ( btp <= 0.001 ) {
  to_add <- "***"
} else if ( btp <= 0.01 ) {
  to_add <- "**"
} else if ( btp <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "n.s."
}
text(50,ymax/10*6.5,labels = to_add, cex=2.5, col = yellows[4])

}

#dev.off()

```



### 0.5.9 Figure S2D

```

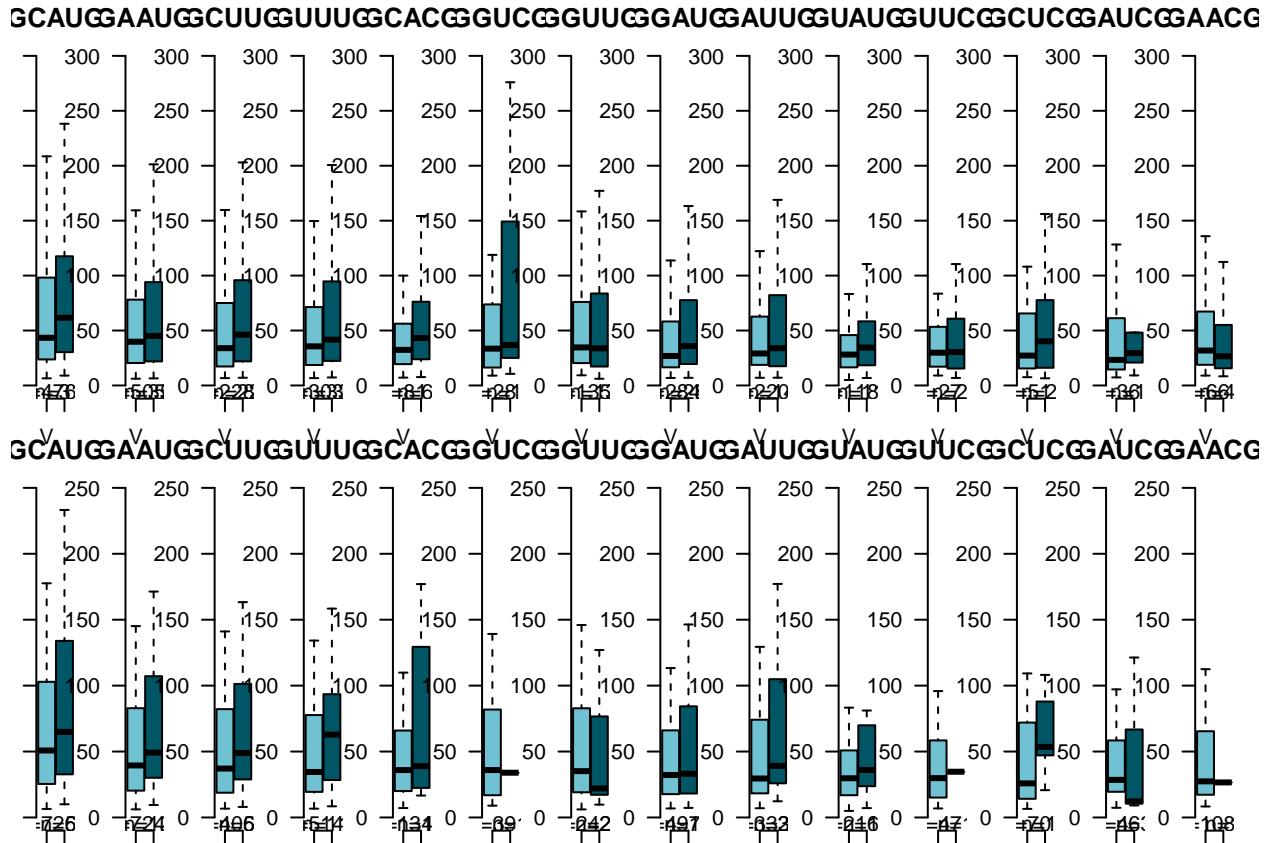
GNWYG <- c("GCATG", "GAATG", "GGATG", "GTTTG", "GCTTG", "GATTG", "GGTTG", "GAACG", "GCACG", "GTATG", "GATCG", "GCT
GNWYG <- GNWYG[order(-medians)]

#pdf("./Figure/Figure2/FigureS2D.pdf", width = 18, height = 5.3)
par(mar=c(1,1,2,1),mfrow=c(2,14),bty="n")

for (i in GNWYG){
  RAPs_GNWYG <- RAPs[["RBFOX2"]][grep(i,str_sub(RAPs[["RBFOX2"]]$positive_fa,50,150)],]
  GNWYG_subset <- GNWYG[grep(i,GNWYG,invert=T)]
  RAPs_GNWYG <- RAPs_GNWYG[grep(paste(GNWYG_subset,collapse = "|"),str_sub(RAPs_GNWYG$positive_fa,50,150))]
  aaa <- paste("T",i,sep="")
  TGNWYG <- RAPs_GNWYG[grep(aaa,str_sub(RAPs_GNWYG$positive_fa,50,150)),]$BS
  VGNWYG <- RAPs_GNWYG[grep(aaa,str_sub(RAPs_GNWYG$positive_fa,50,150), invert = T),]$BS
  NGNWYG <- list(VGNWYG, TGNWYG)
  names(NGNWYG) <- c("V","U")
  boxplot2(NGNWYG, outline=F, ylab=NA, main=gsub("T","U",i), cex.lab=2, las=1, boxwex=0.95,ylim=c(0,300))
}

for (i in GNWYG){
  RAPs_GNWYG <- RAPs[["RBFOX2"]][grep(i,str_sub(RAPs[["RBFOX2"]]$positive_fa,50,150)],]
  GNWYG_subset <- GNWYG[grep(i,GNWYG,invert=T)]
  RAPs_GNWYG <- RAPs_GNWYG[grep(paste(GNWYG_subset,collapse = "|"),str_sub(RAPs_GNWYG$positive_fa,50,150))]
  RAPs_GNWYG$GNWYG_subset <- str_count(str_sub(RAPs_GNWYG$positive_fa,50,150),i)
  RAPs_GNWYG$GNWYG_subset[RAPs_GNWYG$GNWYG_subset>2] <- 2
  RAPs_GNWYG$GNWYG_subset <- factor(as.character(RAPs_GNWYG$GNWYG_subset),levels=c("1","2"))
  boxplot2(data=RAPs_GNWYG,BS~GNWYG_subset, outline=F, ylab=NA,main=gsub("T","U",i), cex.lab=2, las=1, boxwex=0.95,ylim=c(0,300))
}

```



```
#dev.off()
```

### 0.5.10 Figure S2E

```
## the following lines exclude canonical GCAUG as part of the GNWYG to demonstrate that GNWYG acts independently
#pdf("./Figure/Figure2/FigureS2E.pdf", width = 8, height = 5)
par(mfrow=c(1,2),pty="n")
GNWYG <- c("GAATG", "GGATG", "GTTTG", "GCTTG", "GATTG", "GGTTG", "GAACG", "GCACG", "GTATG", "GATCG", "GCTCG", "GTTA")
RAPS_GNWYG_count <- str_count(str_sub(RAPS[["RBFOX2"]])$positive_fa, 50, 150), paste(GNWYG, collapse=" | "))
RAPS_GNWYG_Mean_FCI <- log2(RAPS[["RBFOX2"]][RAPS_GNWYG_count>0, "Mean_FCI"])

RAPS_GNWYG_count[RAPS_GNWYG_count>4] <- 4
RAPS_GNWYG_count <- RAPS_GNWYG_count[RAPS_GNWYG_count>0]
boxplot2(RAPS_GNWYG_Mean_FCI~RAPS_GNWYG_count, outline=F, boxwex=0.95, ylab="Fold Change (log2)", ylim=c(0, 10))

x <- RAPS_GNWYG_Mean_FCI[RAPS_GNWYG_count == 1]
y <- RAPS_GNWYG_Mean_FCI[RAPS_GNWYG_count == 2]
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
```

```

    to_add <- "***"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  a <- "P ="
  b <- round(rst,3)
  to_add <- paste(a,b,sep=" ")
}
text(1.5,7.25,labels = to_add, cex=1.5)

x <- RAPs_GNWYG_Mean_FCI[RAPs_GNWYG_count == 2]
y <- RAPs_GNWYG_Mean_FCI[RAPs_GNWYG_count == 3]
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  a <- "P ="
  b <- round(rst,3)
  to_add <- paste(a,b,sep=" ")
}
text(2.5,7.5,labels = to_add, cex=1.5)

x <- RAPs_GNWYG_Mean_FCI[RAPs_GNWYG_count == 3]
y <- RAPs_GNWYG_Mean_FCI[RAPs_GNWYG_count == 4]
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  a <- "P ="
  b <- round(rst,3)
  to_add <- paste(a,b,sep=" ")
}
text(3.5,7.75,labels = to_add, cex=1.5)

CLIPs_GNWYG_count <- str_count(str_sub(CLIPs[["RBFOX2"]])$positive_fa,20,120),paste(GNWYG,collapse="|"))
CLIPs_GNWYG_MeanFC <- CLIPs[["RBFOX2"]][CLIPs_GNWYG_count>0,"MeanFC"]
CLIPs_GNWYG_count[CLIPs_GNWYG_count>4] <- 4
CLIPs_GNWYG_count <- CLIPs_GNWYG_count[CLIPs_GNWYG_count>0]
boxplot2(CLIPs_GNWYG_MeanFC~CLIPs_GNWYG_count,outline=F,boxwex=0.95, ylab="Fold Change (log2)", ylim=c(0,10))

x <- CLIPs_GNWYG_MeanFC[CLIPs_GNWYG_count == 1]
y <- CLIPs_GNWYG_MeanFC[CLIPs_GNWYG_count == 2]
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
}

```

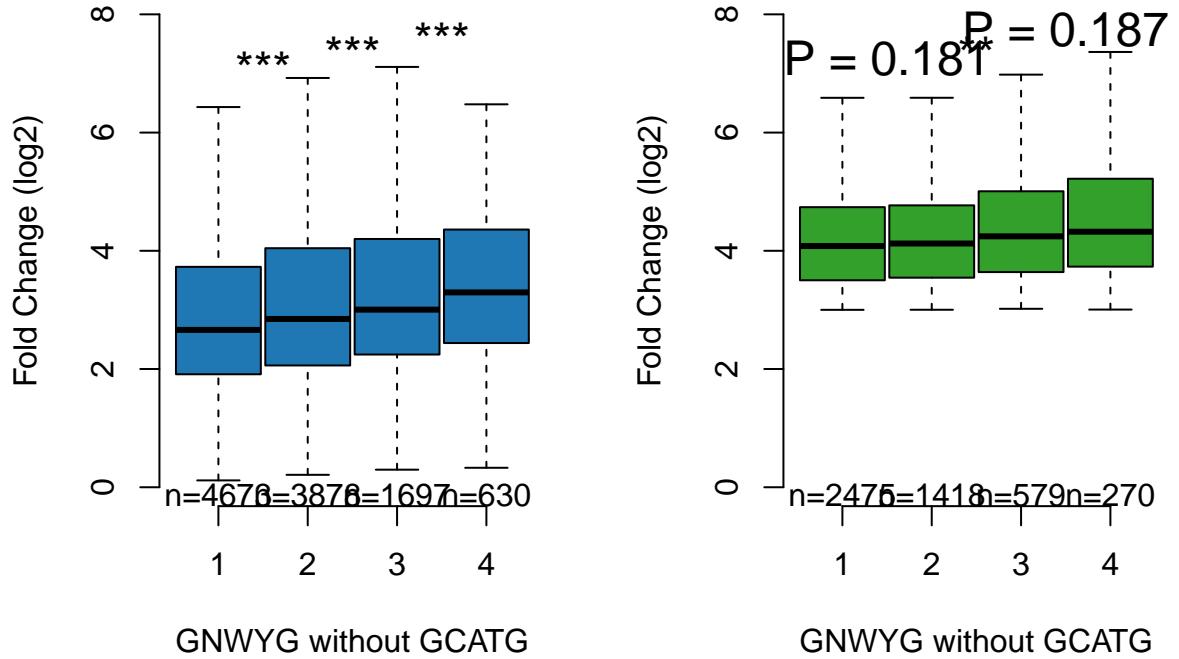
```

} else if ( rst <= 0.01 ) {
  to_add <- "***"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  a <- "P ="
  b <- round(rst,3)
  to_add <- paste(a,b,sep=" ")
}
text(1.5,7.25,labels = to_add, cex=1.5)

x <- CLIPs_GNWYG_MeanFC[CLIPs_GNWYG_count == 2]
y <- CLIPs_GNWYG_MeanFC[CLIPs_GNWYG_count == 3]
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "****"
} else if ( rst <= 0.01 ) {
  to_add <- "***"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  a <- "P ="
  b <- round(rst,3)
  to_add <- paste(a,b,sep=" ")
}
text(2.5,7.5,labels = to_add, cex=1.5)

x <- CLIPs_GNWYG_MeanFC[CLIPs_GNWYG_count == 3]
y <- CLIPs_GNWYG_MeanFC[CLIPs_GNWYG_count == 4]
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "****"
} else if ( rst <= 0.01 ) {
  to_add <- "***"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  a <- "P ="
  b <- round(rst,3)
  to_add <- paste(a,b,sep=" ")
}
text(3.5,7.75,labels = to_add, cex=1.5)

```



```
#dev.off()
```

### 0.5.11 Figure S2F

```

AACAHC <- c("AACATC", "AACACC", "AACAAC")
ACAHC <- c("ACATC", "ACACC", "ACAAC")
CAHC <- c("CATC", "CACC", "CAAC")

AACAHC_scores <- list()
for (i in 1:3){
  RAPs_AACAHC <- RAPs[["YBX3"]][grep(AACAHC[i], str_sub(RAPs[["YBX3"]]$positive_fa, 50, 150)),]
  RAPs_AACAHC_target <- CAHC[-i]
  AACAHC_scores[[i]] <- RAPs_AACAHC[grep(paste(RAPs_AACAHC_target, collapse = "|"), str_sub(RAPs_AACAHC$positive_fa, 50, 150)))]
}
names(AACAHC_scores) <- AACAHC

ACAHC_scores <- list()
for (i in 1:3){
  RAPs_ACAHC <- RAPs[["YBX3"]][grep(ACAHC[i], str_sub(RAPs[["YBX3"]]$positive_fa, 50, 150), invert=T),]
  RAPs_ACAHC <- RAPs_ACAHC[grep(ACAHC[i], str_sub(RAPs_ACAHC$positive_fa, 50, 150)))]
  RAPs_ACAHC_target <- CAHC[-i]
  ACAHC_scores[[i]] <- RAPs_ACAHC[grep(paste(RAPs_ACAHC_target, collapse = "|"), str_sub(RAPs_ACAHC$positive_fa, 50, 150)))]
}
```

```

    ACAHC_scores[[i]] <- RAPs_ACAHC$BS
}
names(ACAHC_scores) <- ACAHC

CAHC_scores <- list()
for (i in 1:3){
  RAPs_CAHC <- RAPs[["YBX3"]][grep(AACAHC[i],str_sub(RAPs[["YBX3"]])$positive_fa,50,150), invert=T,]
  RAPs_CAHC <- RAPs_CAHC[grep(ACAHC[i],str_sub(RAPs_CAHC$positive_fa,50,150), invert=T,]
  RAPs_CAHC <- RAPs_CAHC[grep(CAHC[i],str_sub(RAPs_CAHC$positive_fa,50,150)),]
  RAPs_CAHC_target <- CAHC[-i]
  CAHC_scores[[i]] <- RAPs_CAHC[grep(paste(RAPs_CAHC_target,collapse = " | "),str_sub(RAPs_CAHC$positive_fa,50,150)),]
  CAHC_scores[[i]] <- RAPs_CAHC$BS
}
names(CAHC_scores) <- CAHC

ALL_CAHCs_scores <- c(AACAHC_scores[1],ACAHC_scores[1],CAHC_scores[1], AACAHC_scores[2],ACAHC_scores[2])

#pdf("./Figure/Figure2/FigureS2F.pdf", width = 8, height = 5)
par(bty="n")

boxplot2(ALL_CAHCs_scores,outline=F, range=1, col= c("#1f78b4", "#6baed6", "#c6dbef"), las=3, ylim=c(0,8))

x <- ALL_CAHCs_scores$AACATC
y <- ALL_CAHCs_scores$ACATC
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "ns"
}
text(1.5,77.5,labels = to_add, cex=1.5)

x <- ALL_CAHCs_scores$ACATC
y <- ALL_CAHCs_scores$CATC
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "ns"
}
text(2.5,75,labels = to_add, cex=1.5)

```

```

x <- ALL_CAHCs_scores$AACACC
y <- ALL_CAHCs_scores$ACACC
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "ns"
}
text(4.5,77.5,labels = to_add, cex=1.5)

x <- ALL_CAHCs_scores$ACACC
y <- ALL_CAHCs_scores$CACC
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "ns"
}
text(5.5,75,labels = to_add, cex=1.5)

x <- ALL_CAHCs_scores$AACAC
y <- ALL_CAHCs_scores$ACAC
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {
  to_add <- "ns"
}
text(7.5,77.5,labels = to_add, cex=1.5)

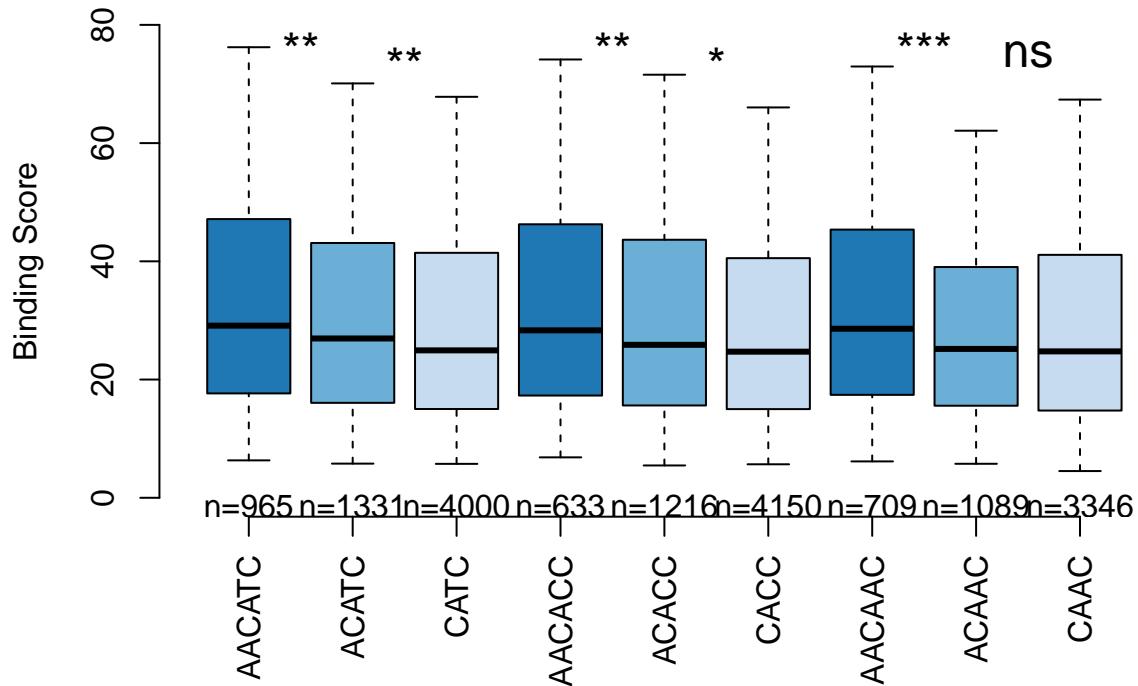
x <- ALL_CAHCs_scores$ACAC
y <- ALL_CAHCs_scores$CAAC
rst <- wilcox.test(x,y)$p.value
if ( rst <= 0.001 ) {
  to_add <- "***"
} else if ( rst <= 0.01 ) {
  to_add <- "**"
} else if ( rst <= 0.05 ) {
  to_add <- "*"
} else {

```

```

    to_add <- "ns"
}
text(8.5,75,labels = to_add, cex=1.5)

```



```
#dev.off()
```

### 0.5.12 Figure S2G

```

Names_protein <- c("HNRNPA1", "HNRNPC", "PTBP1", "RBFOX2", "YBX3", "IGF2BP1")

# CLIPs
CLIPs_RBFOX2_feature_count <- table(CLIPs[["RBFOX2"]][, "feature"])

for (i in Names_protein){
  CLIPs_RBFOX2_feature_tmp <- table(CLIPs[[i]][, "feature"])/nrow(CLIPs[[i]])
  CLIPs_RBFOX2_feature_count <- rbind(CLIPs_RBFOX2_feature_count, CLIPs_RBFOX2_feature_tmp)
}
CLIPs_RBFOX2_feature_count <- CLIPs_RBFOX2_feature_count[-1,]
CLIPs_RBFOX2_feature_count <- t(CLIPs_RBFOX2_feature_count)
colnames(CLIPs_RBFOX2_feature_count) <- Names_protein

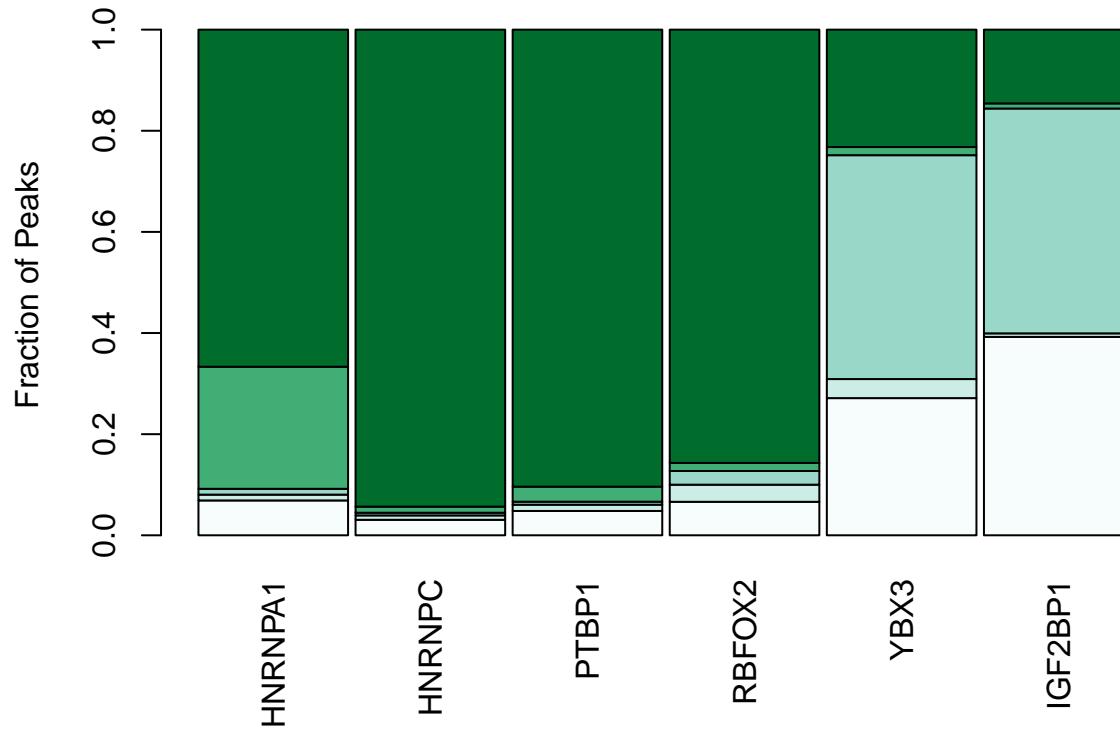
```

```

# RAPs
RAPs_RBFOX2_feature_count <- table(RAPs[["RBFOX2"]][, "feature"])
for (i in Names_protein){
  print(i)
  RAPs_RBFOX2_feature_tmp <- table(RAPs[[i]][, "feature"])/nrow(RAPs[[i]])
  RAPs_RBFOX2_feature_count <- rbind(RAPs_RBFOX2_feature_count, RAPs_RBFOX2_feature_tmp)
}
## [1] "HNRNPA1"
## [1] "HNRNPC"
## [1] "PTBP1"
## [1] "RBFOX2"
## [1] "YBX3"
## [1] "IGF2BP1"
RAPs_RBFOX2_feature_count <- RAPs_RBFOX2_feature_count[-1,]
RAPs_RBFOX2_feature_count <- t(RAPs_RBFOX2_feature_count)
colnames(RAPs_RBFOX2_feature_count) <- Names_protein

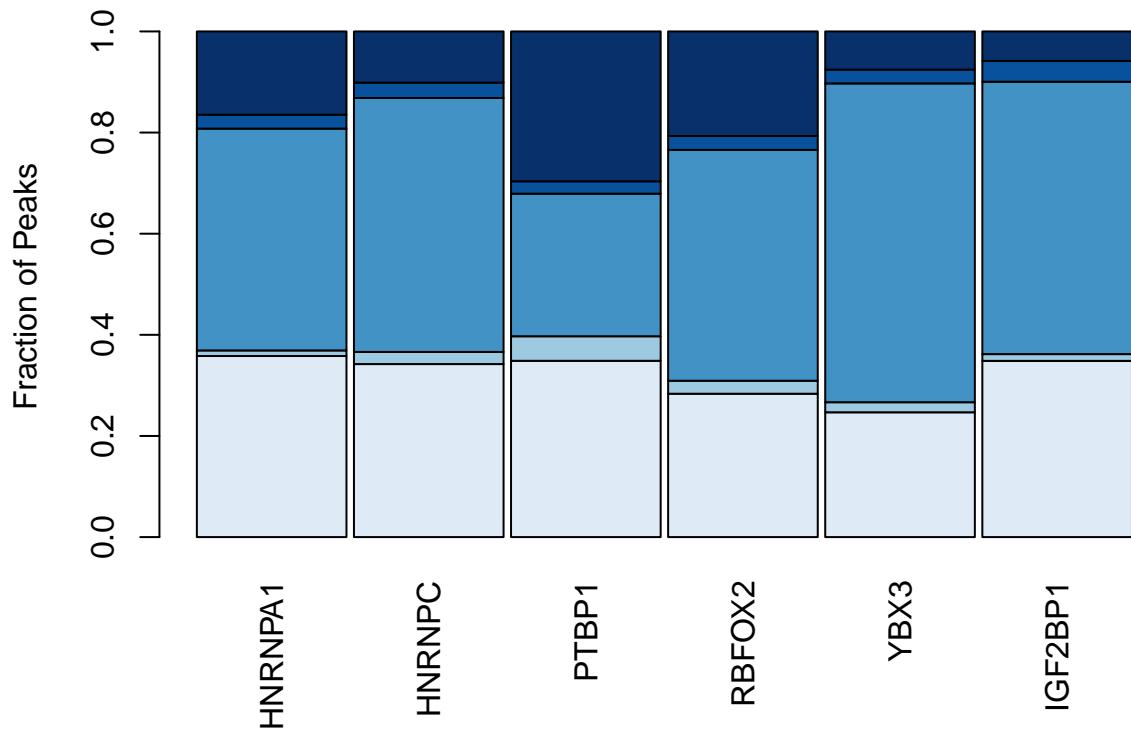
barplot(CLIPs_RBFOX2_feature_count, col=c("#f7fcfd", "#ccece6", "#99d8c9", "#41ae76", "#006d2c"), las=3)

```

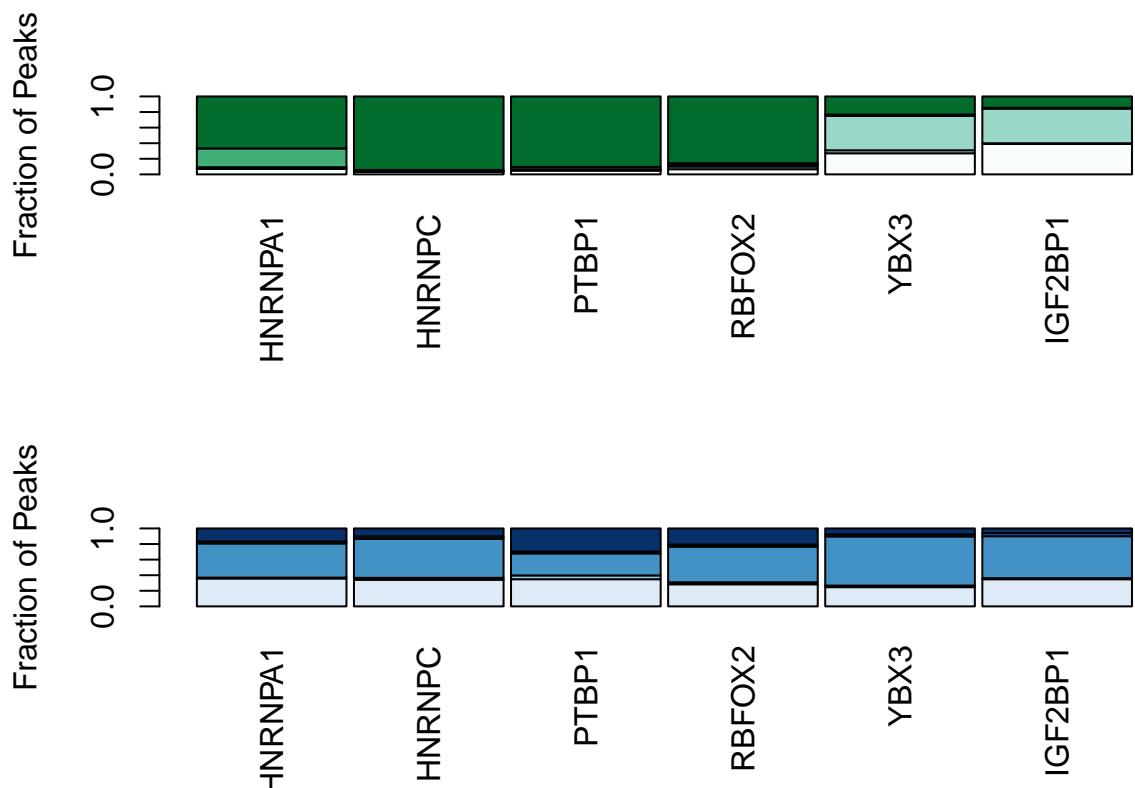


#### 0.5.12.1 include exon

```
barplot(RAPs_RBFOX2_feature_count, col=c("#deebf7", "#9ecae1", "#4292c6", "#08519c", "#08306b"), las=3, spc
```



```
#pdf("./Figure/Figure2/FigureS2G.pdf", width = 13.9, height = 13.7)
par(mfrow=c(2,1))
barplot(CLIPs_RBFOX2_feature_count, col=c("#f7fcfd", "#ccece6", "#99d8c9", "#41ae76", "#006d2c"), las=3)
barplot(RAPs_RBFOX2_feature_count, col=c("#deebf7", "#9ecae1", "#4292c6", "#08519c", "#08306b"), las=3, spc
```



```
#dev.off()
```

```
Names_protein <- c("HNRNPA1", "HNRNPC", "PTBP1", "RBFOX2", "YBX3", "IGF2BP1")

# CLIPs
CLIPs_RBFOX2_feature_count <- table(CLIPs[["RBFOX2"]][CLIPs[["RBFOX2"]]$feature != "exon", "feature"])

for (i in Names_protein){
  CLIPs_RBFOX2_feature_tmp <- table(CLIPs[[i]][CLIPs[[i]]$feature != "exon", "feature"])/nrow(CLIPs[[i]])
  CLIPs_RBFOX2_feature_count <- rbind(CLIPs_RBFOX2_feature_count, CLIPs_RBFOX2_feature_tmp)
}
CLIPs_RBFOX2_feature_count <- CLIPs_RBFOX2_feature_count[-1,]
CLIPs_RBFOX2_feature_count <- t(CLIPs_RBFOX2_feature_count)
colnames(CLIPs_RBFOX2_feature_count) <- Names_protein

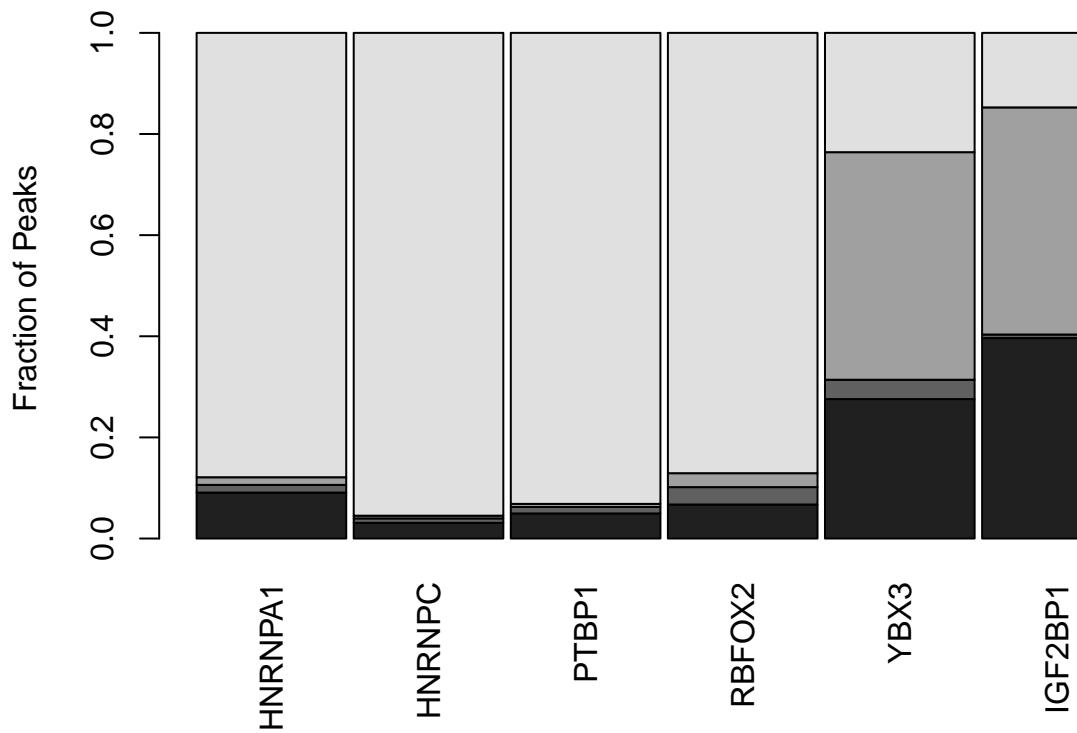
# RAPs
RAPs_RBFOX2_feature_count <- table(RAPs[["RBFOX2"]][RAPs[["RBFOX2"]]$feature != "exon", "feature"])
for (i in Names_protein){
  print(i)
  RAPs_RBFOX2_feature_tmp <- table(RAPs[[i]][RAPs[[i]]$feature != "exon", "feature"])/nrow(RAPs[[i]][RAPs[[i]]$feature != "exon", "feature"])
  RAPs_RBFOX2_feature_count <- rbind(RAPs_RBFOX2_feature_count, RAPs_RBFOX2_feature_tmp)
}
## [1] "HNRNPA1"
```

```

## [1] "HNRNPC"
## [1] "PTBP1"
## [1] "RBFOX2"
## [1] "YBX3"
## [1] "IGF2BP1"
RAPs_RBFOX2_feature_count <- RAPs_RBFOX2_feature_count[-1,]
RAPs_RBFOX2_feature_count <- t(RAPs_RBFOX2_feature_count)
colnames(RAPs_RBFOX2_feature_count) <- Names_protein

barplot(CLIPs_RBFOX2_feature_count, col=greys[c(1,3,5,7)], las=3, space=0.05, ylab="Fraction of Peaks")

```

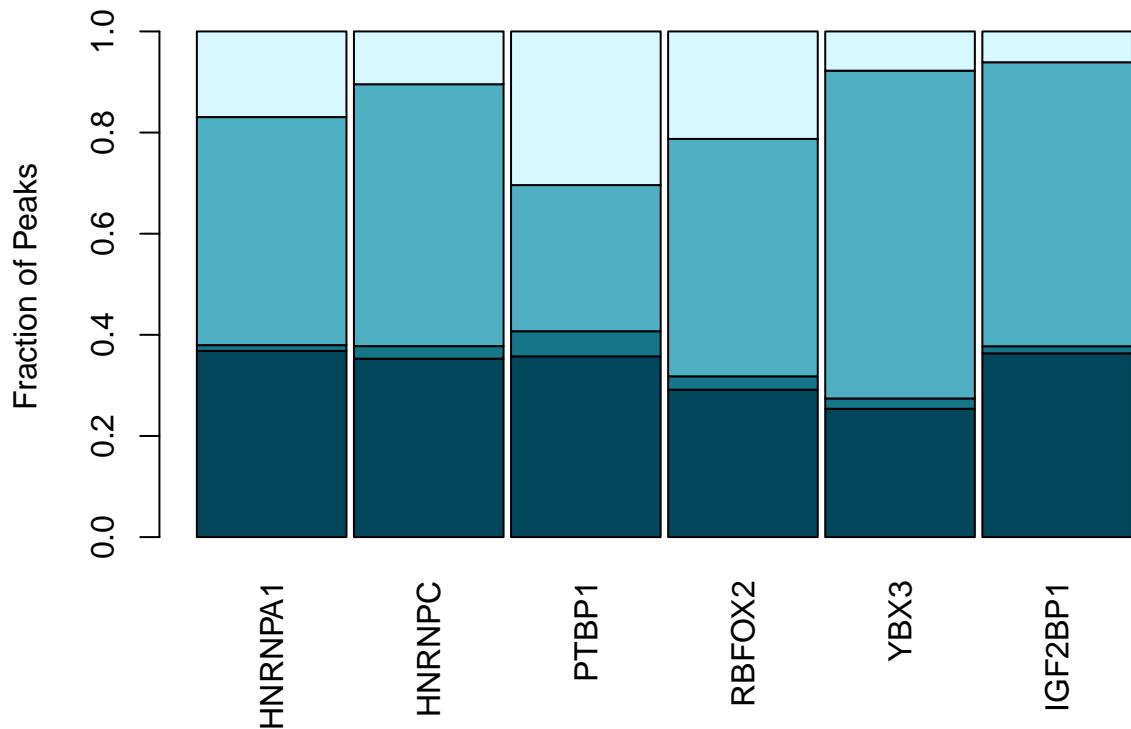


#### 0.5.12.2 exclude exome

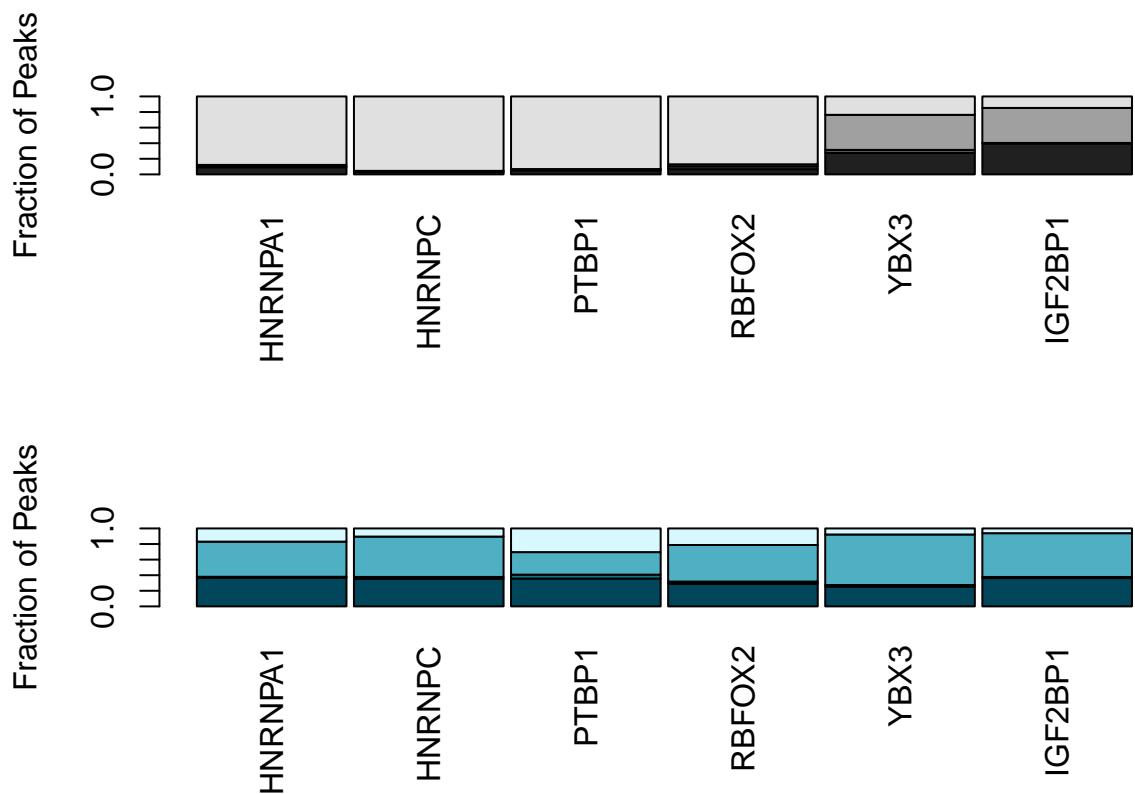
```

barplot(RAPs_RBFOX2_feature_count, col=acqua_blues[c(1,4,7,11)], las=3, space=0.05, ylab="Fraction of Peaks")

```



```
par(mfrow=c(2,1))
barplot(CLIPs_RBFOX2_feature_count, col=greys[c(1,3,5,7)], las=3, space=0.05, ylab="Fraction of Peaks")
barplot(RAPs_RBFOX2_feature_count, col=acqua_blues[c(1,4,7,11)], las=3, space=0.05, ylab="Fraction of Peaks")
```



```

sessionInfo()
## R version 4.3.2 (2023-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS 15.5
##
## Matrix products: default
## BLAS:    /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK:  /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib; LAPACK v
## 
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
## 
## time zone: Europe/Stockholm
## tzcode source: internal
## 
## attached base packages:
## [1] grid      stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
## 
## other attached packages:
## [1] zoo_1.8-12                  robustbase_0.99-4
## [3] ggseqlogo_0.2                euler_7.0.2
## [5] GGally_2.2.1                 idr_1.3
## [7] ggforce_0.4.2                corrplot_0.94
## [9] phastCons100way.UCSC.hg38_3.7.1 phastCons7way.UCSC.hg38_3.7.1
## [11] GenomicScores_2.14.3        ggpibr_0.6.0

```

```

## [13] UpSetR_1.4.0
## [15] LSD_4.1-0
## [17] futile.logger_1.4.3
## [19] ggbeeswarm_0.7.2
## [21] gtools_3.9.5
## [23] org.Hs.eg.db_3.18.0
## [25] gplots_3.1.3.1
## [27] AnnotationDbi_1.64.1
## [29] reshape2_1.4.4
## [31] dendextend_1.17.1
## [33] lubridate_1.9.3
## [35] stringr_1.5.1
## [37] purrr_1.0.2
## [39] tidyverse_1.3.1
## [41] tidyverse_2.0.0
## [43] ggplot2_3.5.1
## [45] limma_3.58.1
## [47] XVector_0.42.0
## [49] GenomicRanges_1.54.1
## [51] IRanges_2.36.0
## [53] BiocGenerics_0.48.1
##
## loaded via a namespace (and not attached):
##   [1] fs_1.6.4
##   [3] bitops_1.0-8
##   [5] HDO.db_0.99.1
##   [7] RColorBrewer_1.1-3
##   [9] tools_4.3.2
##  [11] utf8_1.2.4
##  [13] HDF5Array_1.30.1
##  [15] lazyeval_0.2.2
##  [17] withr_3.0.1
##  [19] prettyunits_1.2.0
##  [21] cli_3.6.3
##  [23] scatterpie_0.2.4
##  [25] systemfonts_1.1.0
##  [27] yulab.utils_0.1.7
##  [29] DOSE_3.28.2
##  [31] rstudioapi_0.16.0
##  [33] generics_0.1.3
##  [35] BiocIO_1.12.0
##  [37] GO.db_3.18.0
##  [39] fansi_1.0.6
##  [41] lifecycle_1.0.4
##  [43] carData_3.0-5
##  [45] rhdf5_2.46.1
##  [47] SparseArray_1.2.4
##  [49] blob_1.2.4
##  [51] crayon_1.5.3
##  [53] cowplot_1.1.3
##  [55] pillar_1.9.0
##  [57] fgsea_1.28.0
##  [59] codetools_0.2-20
##   [1] gridExtra_2.3
##   [3] VennDiagram_1.7.3
##   [5] ReactomePA_1.46.0
##   [7] ggrepel_0.9.5
##   [9] org.Dr eg.db_3.18.0
##  [11] clusterProfiler_4.10.1
##  [13] GenomicFeatures_1.54.4
##  [15] Biobase_2.62.0
##  [17] dichromat_2.0-0.1
##  [19] scales_1.3.0
##  [21] forcats_1.0.0
##  [23] dplyr_1.1.4
##  [25] readr_2.1.5
##  [27] tibble_3.2.1
##  [29] ggfortify_0.4.17
##  [31] edgeR_4.0.16
##  [33] Biostrings_2.70.3
##  [35] ChIPpeakAnno_3.36.1
##  [37] GenomeInfoDb_1.38.8
##  [39] S4Vectors_0.40.2
##   [1] matrixStats_1.3.0
##   [3] enrichplot_1.22.0
##   [5] httr_1.4.7
##   [7] InteractionSet_1.30.0
##   [9] backports_1.5.0
##  [11] R6_2.5.1
##  [13] mgcv_1.9-1
##  [15] rhdf5filters_1.14.1
##  [17] graphite_1.48.0
##  [19] textshaping_0.4.0
##  [21] formatR_1.14
##  [23] labeling_0.4.3
##  [25] Rsamtools_2.18.0
##  [27] gson_0.1.0
##  [29] BSgenome_1.70.2
##  [31] RSQLite_2.3.7
##  [33] gridGraphics_0.5-1
##  [35] car_3.1-2
##  [37] Matrix_1.6-5
##  [39] abind_1.4-5
##  [41] yaml_2.3.10
##  [43] SummarizedExperiment_1.32.0
##  [45] qvalue_2.34.0
##  [47] BiocFileCache_2.10.2
##  [49] promises_1.3.0
##  [51] lattice_0.22-6
##  [53] KEGGREST_1.42.0
##  [55] knitr_1.48
##  [57] rjson_0.2.21
##  [59] fastmatch_1.1-4

```

```

## [61] glue_1.7.0                      gggfun_0.1.6
## [63] data.table_1.16.0                vctrs_0.6.5
## [65] png_0.1-8                       treeio_1.26.0
## [67] gtable_0.3.5                     cachem_1.1.0
## [69] xfun_0.47                       mime_0.12
## [71] S4Arrays_1.2.1                  tidygraph_1.3.1
## [73] survival_3.7-0                 statmod_1.5.0
## [75] interactiveDisplayBase_1.40.0   nlme_3.1-166
## [77] ggtree_3.10.1                   bit64_4.0.5
## [79] progress_1.2.3                  filelock_1.0.3
## [81] viper_0.4.7                     KernSmooth_2.23-24
## [83] colorspace_2.1-1                DBI_1.2.3
## [85] tidyselect_1.2.1                bit_4.0.5
## [87] compiler_4.3.2                 curl_5.2.2
## [89] graph_1.80.0                   xml2_1.3.6
## [91] DelayedArray_0.28.0             shadowtext_0.1.4
## [93] rtracklayer_1.62.0              caTools_1.18.2
## [95] DEoptimR_1.1-3                 RBGL_1.78.0
## [97] rappdirs_0.3.3                 digest_0.6.37
## [99] rmarkdown_2.28                  htmltools_0.5.8.1
## [101] pkgconfig_2.0.3                MatrixGenerics_1.14.0
## [103] highr_0.11                     dbplyr_2.5.0
## [105] regioneR_1.34.0               fastmap_1.2.0
## [107] rlang_1.1.4                   shiny_1.9.1
## [109] farver_2.1.2                  jsonlite_1.8.8
## [111] BiocParallel_1.36.0            GOSemSim_2.28.1
## [113] RCurl_1.98-1.16               magrittr_2.0.3
## [115] GenomeInfoDbData_1.2.11      ggplotify_0.1.2
## [117] patchwork_1.2.0                Rhdf5lib_1.24.2
## [119] munsell_0.5.1                 Rcpp_1.0.13
## [121] ape_5.8                       viridis_0.6.5
## [123] stringi_1.8.4                ggraph_2.2.1
## [125] zlibbioc_1.48.2               MASS_7.3-60.0.1
## [127] AnnotationHub_3.10.1          plyr_1.8.9
## [129] ggstats_0.6.0                 parallel_4.3.2
## [131] graphlayouts_1.1.1             splines_4.3.2
## [133] multtest_2.58.0               hms_1.1.3
## [135] polylabelr_0.2.0              locfit_1.5-9.10
## [137] igraph_2.0.3                 ggsignif_0.6.4
## [139] biomaRt_2.58.2               futile.options_1.0.1
## [141] BiocVersion_3.18.1            XML_3.99-0.17
## [143] evaluate_0.24.0               lambda.r_1.2.4
## [145] BiocManager_1.30.25           tzdb_0.4.0
## [147] tweenr_2.0.3                 httpuv_1.6.15
## [149] polyclip_1.10-7              xtable_1.8-4
## [151] broom_1.0.6                  restfulr_0.0.15
## [153] reactome.db_1.86.2            tidytree_0.4.6
## [155] rstatix_0.7.2                later_1.3.2
## [157] ragg_1.3.2                   viridisLite_0.4.2
## [159] aplot_0.2.3                  memoise_2.0.1
## [161] beeswarm_0.4.0                GenomicAlignments_1.38.2
## [163] timechange_0.3.0

```