irCLIPv2 and Re-CLIP dataset from EGF timecourse for HNRNPC and UPF1

Luca Ducoli

This is the pipeline used to analyze the irCLIPv2 and Re-CLIP datasets of UPF1 and HNRNPC from EGF time course. HNRNPC and UPF1 irCLIPv2 was performed at 0, 15, 30, and 60min EGF stimulation. HNRNPC and UPF1 irCLIPv2 after the knockdown of the UPF1 and HNRNPC were performed at 0 and 60min EGF stimulation. HNRNPC-UPF1 RE-CLIP was performed at 0, 30, 60min EGF stimualtion. The experiments were performed in duplicates for each time point in A431 cells.

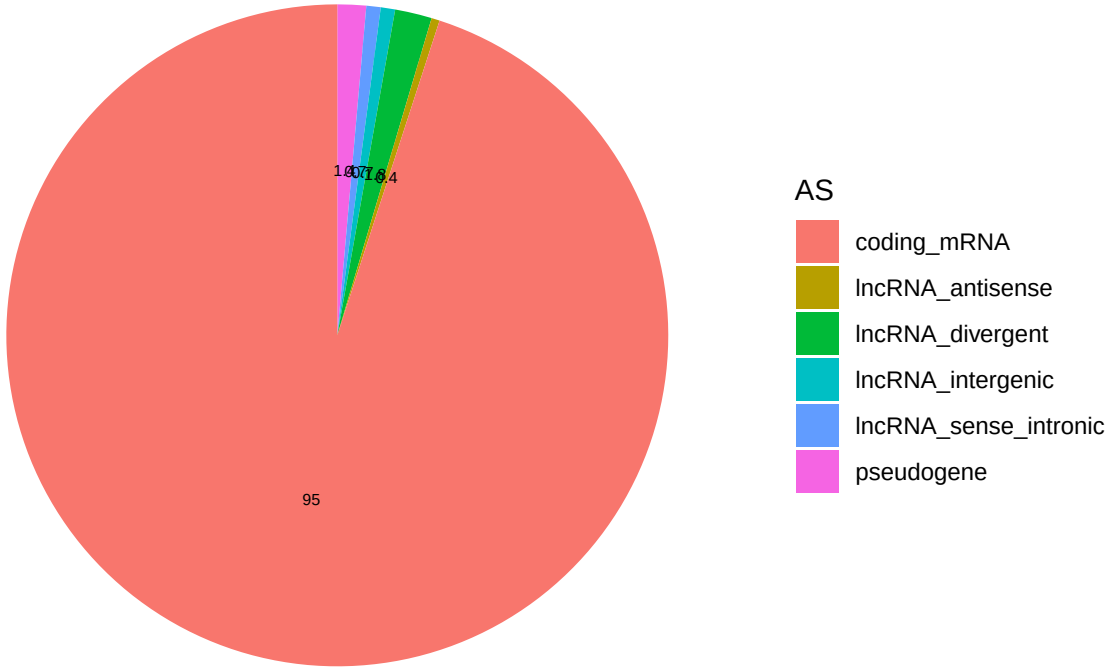
1. Characterization of RI events

We first characterize the RI events in terms of the genomic location.

```
library(tidyverse)
library(Repitools)
library(AnnotationHub)
library(AnnotationDbi)
library(ggplot2)
library(plyranges)
library(GenomicRanges)
library(gintools)
library(rtracklayer)
library(EnrichedHeatmap)
library(circlize)
library(caTools)
library(cliProfiler)
library(paletteer)
library(gridExtra)
library(NbClust)
library(RColorBrewer)
library(ChIPpeakAnno)
library(data.table)
library(BRGenomics)
library(DESeq2)
library(ggpmisc)
library(eulerr)
library(memes)
library(BSgenome.Hsapiens.UCSC.hg38)
library(universalmotif)
library(SLIMFinderR)
```

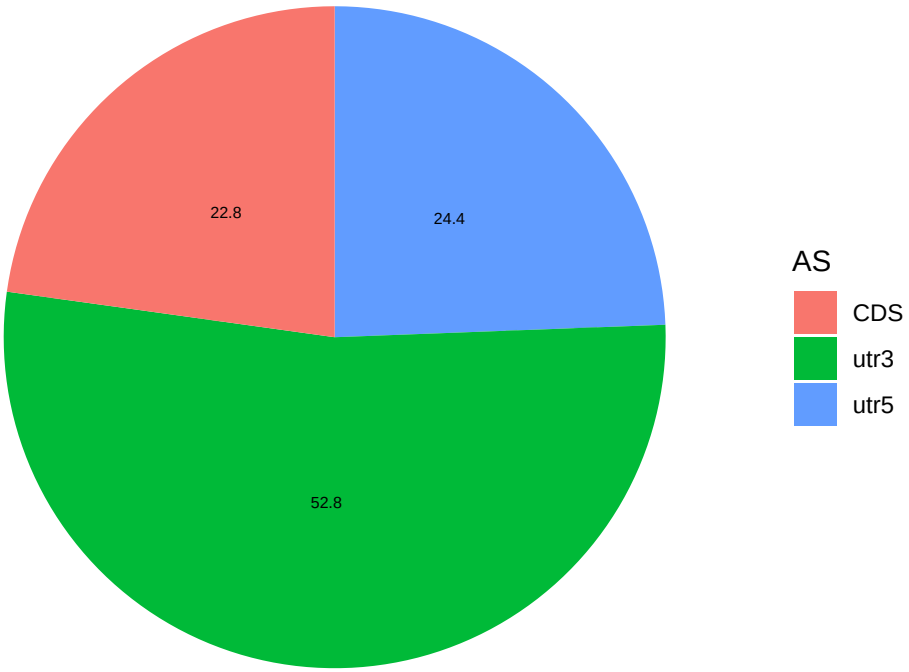
```
#Get transcript names from rMATs output
gtf <- rtracklayer::import('~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/siRNA_EGF_SG/2_EGF/8_
    DTE_analysis/0_UPF1_HNRNPC/1_DTE/Genome/gencode.v39.annotation.gtf')
gtf_df=as.data.frame(gtf)

AS_event <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/siRNA_EGF_SG/2_EGF/8_DTE_
    analysis/0_UPF1_HNRNPC/0_Overlap/HNRNPC_UPF1_RIevents.txt", header = TRUE)
AS_event <- AS_event %>% mutate("gene_id" = sapply(strsplit(AS_event$region, "_"), function(x) x
    [1]),
```

```
                                    "gene_name"= sapply(strsplit(AS_event$region, "_"), function(x) x
    [2]),
                                    "chr"= sapply(strsplit(AS_event$region, "_"), function(x) x[3]),
                                    "strand"= sapply(strsplit(AS_event$region, "_"), function(x) x
    [4]),
                                    "start" = sapply(strsplit(AS_event$region, "_"), function(x) x
    [5]),
                                    "end"= sapply(strsplit(AS_event$region, "_"), function(x) x[6]))
    %>% dplyr::select(c(region,chr,start,end, strand, gene_id, gene_name, AS))

gene.info <- read.delim("/Users/lducoli/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/0
    _Annotation/0_F6_CAT.gene.info.tsv.txt", header = TRUE)

AS_event$geneID <- gsub("\\..*", "", AS_event$gene_id)
AS_event <- merge(AS_event, gene.info[,1:4], by = "geneID", all.x = TRUE)
AS_event$CAT_geneClass[AS_event$geneID == "ENSG00000273559"] <- "coding_mRNA"
AS_event$CAT_geneClass[AS_event$geneID == "ENSG00000288663"] <- "lncRNA_sense_intronic"

#Pie Chart
regionRes <- unique(AS_event %>% dplyr::select(geneID, CAT_geneClass)) %>% dplyr::count(CAT_
    geneClass)
regionRes$AS <- factor(regionRes$CAT_geneClass)

regionRes <- regionRes %>%
  arrange(desc(AS)) %>%
  mutate(prop = round((n / sum(regionRes$n))*100,1)) %>%
  mutate(ypos = cumsum(prop)- 0.5*prop )

piechart <- ggplot(regionRes, aes(x="", y=prop, fill=AS))+ geom_bar(width = 1, stat = "identity")
    +
    coord_polar("y", start=0) +
  theme_minimal()+
  theme(
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  panel.border = element_blank(),
  panel.grid=element_blank(),
  axis.ticks = element_blank(),
  plot.title=element_text(size=14, face="bold")
  )+
  theme(axis.text.x=element_blank()) +
  geom_text(aes(y = ypos, label = prop), color = "black", size=2)

piechart
```




```
#Subset to protein coding
AS_event <- subset(AS_event, CAT_geneClass == "coding_mRNA")
write.table(AS_event, "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/
    1_Distribution/PC_RI_events.txt", quote = F, sep = "\t")

#Pie chart genomic regions
region.gr <- GRanges(AS_event[,grep("chr", colnames(AS_event))], IRanges(start=as.numeric(AS_
    event[,grep("start", colnames(AS_event))]), end=as.numeric(AS_event[,grep("end", colnames(AS_
    event))]), names = AS_event[,grep("region", colnames(AS_event))]), strand = AS_event[,grep("
    strand", colnames(AS_event))])

txdb <- loadDb("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/1_Subzones/0_Annotation/gencode.
    v39.annotation")

dist <- genomicElementDistribution(region.gr, TxDb = txdb, promoterRegion=c(upstream=0,
    downstream=0), geneDownstream=c(upstream=0, downstream=0), ignore.strand = FALSE, plot =
    FALSE)

region_anno <- as.data.frame(mcols(dist$peaks))
region_anno$Exons <- ifelse(region_anno$Exons == "otherExon", "CDS", region_anno$Exons)
regionRes2 <- region_anno %>% dplyr::count(Exons)
regionRes2$AS <- factor(regionRes2$Exons)

regionRes2 <- regionRes2 %>%
  arrange(desc(AS)) %>%
  mutate(prop = round((n / sum(regionRes2$n))*100,1)) %>%
  mutate(ypos = cumsum(prop)- 0.5*prop )

piechart2 <- ggplot(regionRes2, aes(x="", y=prop, fill=AS))+ geom_bar(width = 1, stat = "identity
    ") +
    coord_polar("y", start=0) +
  theme_minimal()+
  theme(
```

```
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  panel.border = element_blank(),
  panel.grid=element_blank(),
  axis.ticks = element_blank(),
  plot.title=element_text(size=14, face="bold")
  )+
  theme(axis.text.x=element_blank()) +
  geom_text(aes(y = ypos, label = prop), color = "black", size=2)

piechart2
```




```
#Get the transcripts
AS_event_tx <- merge(AS_event %>% dplyr::select(region,gene_id,AS), gtf_df[gtf_df$type == "
    transcript",], by = "gene_id") %>% dplyr::select(c(seqnames,start,end,transcript_id, strand,
    gene_id, gene_name, AS))

AS_event_bed <- AS_event %>% dplyr::select(chr, start, end, strand, region, gene_id, AS)
AS_event_tx_bed <- AS_event_tx %>% dplyr::select(seqnames, start, end, strand, transcript_id,
    gene_id)

#bedtools to get regions inside transcripts
AS_event_tx2 <- bedtoolsr::bt.intersect(AS_event_bed, AS_event_tx_bed, loj = TRUE)
AS_event_tx2 <- unique(AS_event_tx2)

colnames(AS_event_tx2) <- c("AS_chr", "AS_start", "AS_end", "AS_strand", "AS_region", "AS_geneID"
    , "AS_event", "TX_chr", "TX_start", "TX_end", "TX_strand", "TX_id", "TX_geneID")
AS_event_tx2 <- AS_event_tx2 %>% mutate(geneIDmatch = case_when(AS_geneID != TX_geneID  ~ "0",AS_
    geneID == TX_geneID ~ "1"))
AS_event_tx2 <- subset(AS_event_tx2, geneIDmatch > 0)
```

```
# Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/1_Distribution/
    Piechart_anno_HNRNPC_UPF1_RI_region.pdf",  height=5, width=5)
piechart
piechart2
dev.off()
```

2. Check the distribution of RT stops for HNRNPC and UPF1 and re-CLIP data

In order to identify where on the RI transcripts the co-binding events between UPF1 and HNRNPC are happening, we analyzed the distribution of the RT stop location across several genomic features (5'UTR, exon, intron 3'UTR).

```
#Get annotation
gff_anno <- rtracklayer::import.gff3('~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/0_
    Annotation/gencode.v39.annotation.gff3')

#Get normalized bigwigfiles
egf_granges <- function(rbp, tp) {
setwd("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/3_Bigwig_sum")
bw.p <- rtracklayer::import.bw(paste(rbp, "A431", tp, "merged.p.scaleddewseq.bw", sep = "_"))
bw.m <- rtracklayer::import.bw(paste(rbp, "A431", tp, "merged.m.scaleddewseq.bw", sep = "_"))

strand(bw.p) <- "+"
strand(bw.m) <- "-"

bw <- bind_ranges(bw.p, bw.m)
bw$timepoint <- gsub("_HMW", "", tp)
bw$rbp <- rbp
bw$score <- NULL
return(bw)
}

bw.EGF0.HNRNPC <- egf_granges("HNRNPC", "EGF0")
bw.EGF15.HNRNPC <- egf_granges("HNRNPC", "EGF15")
bw.EGF30.HNRNPC <- egf_granges("HNRNPC", "EGF30")
bw.EGF60.HNRNPC <- egf_granges("HNRNPC", "EGF60")

bw.EGF0.UPF1 <- egf_granges("UPF1", "EGF0_HMW")
bw.EGF15.UPF1 <- egf_granges("UPF1", "EGF15_HMW")
bw.EGF30.UPF1 <- egf_granges("UPF1", "EGF30_HMW")
bw.EGF60.UPF1 <- egf_granges("UPF1", "EGF60_HMW")

#Get normalized bigwigfiles from reclip
egf_granges <- function(rbp, tp) {
setwd("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/6_reclip-EGF_tcupf1/3_Bigwig_sum")
bw.p <- rtracklayer::import.bw(paste(rbp, "A431_hnC", tp, "merged.p.scaleddewseq.bw", sep = "_"))
bw.m <- rtracklayer::import.bw(paste(rbp, "A431_hnC", tp, "merged.m.scaleddewseq.bw", sep = "_"))

strand(bw.p) <- "+"
strand(bw.m) <- "-"

bw <- bind_ranges(bw.p, bw.m)
bw$timepoint <- gsub("_HMW", "", tp)
bw$rbp <- paste("re", rbp, sep = "_")
bw$score <- NULL
```

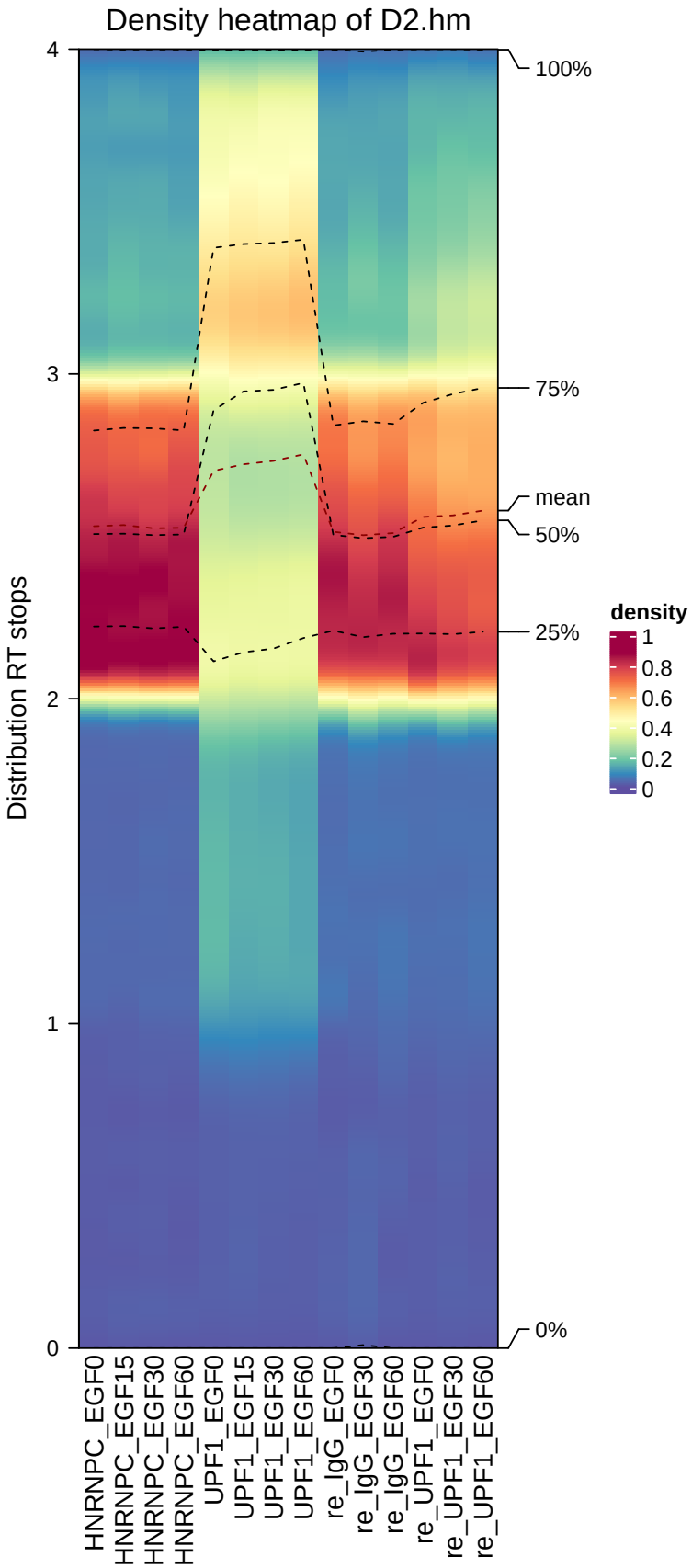
```
return(bw)
}

bw.reEGF0.IgG <- egf_granges("IgG", "EGF0")
bw.reEGF30.IgG <- egf_granges("IgG", "EGF30")
bw.reEGF60.IgG <- egf_granges("IgG", "EGF60")

bw.reEGF0.UPF1 <- egf_granges("UPF1", "EGF0_HMW")
bw.reEGF30.UPF1 <- egf_granges("UPF1", "EGF30_HMW")
bw.reEGF60.UPF1 <- egf_granges("UPF1", "EGF60_HMW")

#Subset to AS transcripts
gr.tx <- GRanges(AS_event_tx2[,grep("^TX_chr$", colnames(AS_event_tx2))],
                 IRanges(start=AS_event_tx2[,grep("^TX_start$", colnames(AS_event_tx2))],
                         end=AS_event_tx2[,grep("^TX_end$", colnames(AS_event_tx2))]),
                 tx_name = AS_event_tx2[,grep("^TX_id$", colnames(AS_event_tx2))],
                 strand = AS_event_tx2[,grep("^TX_strand$", colnames(AS_event_tx2))],
                 gene_name = AS_event_tx2[,grep("^TX_geneID$", colnames(AS_event_tx2))],
                width_region = AS_event_tx2[,grep("^TX_end$", colnames(AS_event_tx2))]-AS_event_
    tx2[,grep("^TX_start$", colnames(AS_event_tx2))],
                AS = AS_event_tx2[,grep("^AS_event$", colnames(AS_event_tx2))])
gr.tx <- unique_granges(gr.tx)


bw.EGF0.HNRNPC <- find_overlaps(bw.EGF0.HNRNPC, gr.tx)
mcols(bw.EGF0.HNRNPC) <- mcols(bw.EGF0.HNRNPC)[c(1,2,6)]
bw.EGF0.HNRNPC <- unique_granges(bw.EGF0.HNRNPC)

bw.EGF15.HNRNPC <- find_overlaps(bw.EGF15.HNRNPC, gr.tx)
mcols(bw.EGF15.HNRNPC) <- mcols(bw.EGF15.HNRNPC)[c(1,2,6)]
bw.EGF15.HNRNPC <- unique_granges(bw.EGF15.HNRNPC)

bw.EGF30.HNRNPC <- find_overlaps(bw.EGF30.HNRNPC, gr.tx)
mcols(bw.EGF30.HNRNPC) <- mcols(bw.EGF30.HNRNPC)[c(1,2,6)]
bw.EGF30.HNRNPC <- unique_granges(bw.EGF30.HNRNPC)

bw.EGF60.HNRNPC <- find_overlaps(bw.EGF60.HNRNPC, gr.tx)
mcols(bw.EGF60.HNRNPC) <- mcols(bw.EGF60.HNRNPC)[c(1,2,6)]
bw.EGF60.HNRNPC <- unique_granges(bw.EGF60.HNRNPC)

bw.EGF0.UPF1 <- find_overlaps(bw.EGF0.UPF1, gr.tx)
mcols(bw.EGF0.UPF1) <- mcols(bw.EGF0.UPF1)[c(1,2,6)]
bw.EGF0.UPF1 <- unique_granges(bw.EGF0.UPF1)

bw.EGF15.UPF1 <- find_overlaps(bw.EGF15.UPF1, gr.tx)
mcols(bw.EGF15.UPF1) <- mcols(bw.EGF15.UPF1)[c(1,2,6)]
bw.EGF15.UPF1 <- unique_granges(bw.EGF15.UPF1)

bw.EGF30.UPF1 <- find_overlaps(bw.EGF30.UPF1, gr.tx)
mcols(bw.EGF30.UPF1) <- mcols(bw.EGF30.UPF1)[c(1,2,6)]
bw.EGF30.UPF1 <- unique_granges(bw.EGF30.UPF1)

bw.EGF60.UPF1 <- find_overlaps(bw.EGF60.UPF1, gr.tx)
mcols(bw.EGF60.UPF1) <- mcols(bw.EGF60.UPF1)[c(1,2,6)]
bw.EGF60.UPF1 <- unique_granges(bw.EGF60.UPF1)

bw.reEGF0.IgG <- find_overlaps(bw.reEGF0.IgG, gr.tx)
mcols(bw.reEGF0.IgG) <- mcols(bw.reEGF0.IgG)[c(1,2,6)]
bw.reEGF0.IgG <- unique_granges(bw.reEGF0.IgG)

bw.reEGF30.IgG <- find_overlaps(bw.reEGF30.IgG, gr.tx)
mcols(bw.reEGF30.IgG) <- mcols(bw.reEGF30.IgG)[c(1,2,6)]
bw.reEGF30.IgG <- unique_granges(bw.reEGF30.IgG)

bw.reEGF60.IgG <- find_overlaps(bw.reEGF60.IgG, gr.tx)
mcols(bw.reEGF60.IgG) <- mcols(bw.reEGF60.IgG)[c(1,2,6)]
bw.reEGF60.IgG <- unique_granges(bw.reEGF60.IgG)
```

```
bw.reEGF0.UPF1 <- find_overlaps(bw.reEGF0.UPF1, gr.tx)
mcols(bw.reEGF0.UPF1) <- mcols(bw.reEGF0.UPF1)[c(1,2,6)]
bw.reEGF0.UPF1 <- unique_granges(bw.reEGF0.UPF1)

bw.reEGF30.UPF1 <- find_overlaps(bw.reEGF30.UPF1, gr.tx)
mcols(bw.reEGF30.UPF1) <- mcols(bw.reEGF30.UPF1)[c(1,2,6)]
bw.reEGF30.UPF1 <- unique_granges(bw.reEGF30.UPF1)

bw.reEGF60.UPF1 <- find_overlaps(bw.reEGF60.UPF1, gr.tx)
mcols(bw.reEGF60.UPF1) <- mcols(bw.reEGF60.UPF1)[c(1,2,6)]
bw.reEGF60.UPF1 <- unique_granges(bw.reEGF60.UPF1)

bw.EGF <- bind_ranges(bw.EGF0.HNRNPC, bw.EGF15.HNRNPC, bw.EGF30.HNRNPC, bw.EGF60.HNRNPC,
                      bw.EGF0.UPF1,bw.EGF15.UPF1,bw.EGF30.UPF1,bw.EGF60.UPF1,
                      bw.reEGF0.IgG, bw.reEGF30.IgG, bw.reEGF60.IgG, bw.reEGF0.UPF1, bw.reEGF30.
    UPF1, bw.reEGF60.UPF1)

#UTR5
UTR5 <- gff_anno[gff_anno$type == "five_prime_UTR"]
UTR5_profile <- windowProfile(bw.EGF, UTR5)
UTR5_profile$Peaks$type <- "UTR5"
UTR5_profile$Peaks <- subset(UTR5_profile$Peaks, window_map != 3)

#Exon
exon <- gff_anno[gff_anno$type == "CDS"]
exon_profile <- windowProfile(bw.EGF, exon)
exon_profile$Peaks$type <- "exon"
exon_profile$Peaks <- subset(exon_profile$Peaks, window_map != 3)

#Intron
intron_profile <- intronProfile(bw.EGF, annotation = "~/Documents/Postdoc/PD_Projects/3_irCLIP-
    RNP/Seq/0_Annotation/gencode.v39.annotation_protcoding.gff3")
intron_profile$Peaks$type <- "intron"
intron_profile$Peaks <- subset(intron_profile$Peaks, Intron_map != 3)

#UTR3
UTR3 <- gff_anno[gff_anno$type == "three_prime_UTR"]
UTR3_profile <- windowProfile(bw.EGF, UTR3)
UTR3_profile$Peaks$type <- "UTR3"
UTR3_profile$Peaks <- subset(UTR3_profile$Peaks, window_map != 3)

df <- data.frame("Position" = c(UTR5_profile$Peaks$window_map, exon_profile$Peaks$window_map,
    intron_profile$Peaks$Intron_map, UTR3_profile$Peaks$window_map), "location" = c(UTR5_profile$
    Peaks$type, exon_profile$Peaks$type, intron_profile$Peaks$type, UTR3_profile$Peaks$type), "
    group" = c(UTR5_profile$Peaks$timepoint, exon_profile$Peaks$timepoint, intron_profile$Peaks$
    timepoint, UTR3_profile$Peaks$timepoint), "rbp" = c(UTR5_profile$Peaks$rbp, exon_profile$
    Peaks$rbp, intron_profile$Peaks$rbp, UTR3_profile$Peaks$rbp))

df$Position[df$location == "exon"] <- df$Position[df$location == "exon"] + 1
df$Position[df$location == "intron"] <- df$Position[df$location == "intron"] + 2
df$Position[df$location == "UTR3"] <- df$Position[df$location == "UTR3"] + 3
```

Density heatmap of distribution

```
#Density heatmap
length <- max(c(length(df$Position[df$rbp == "HNRNPC" & df$group == "EGF0"]), length(df$Position[
    df$rbp == "HNRNPC" & df$group == "EGF15"]),
                length(df$Position[df$rbp == "HNRNPC" & df$group == "EGF30"]), length(df$Position
    [df$rbp == "HNRNPC" & df$group == "EGF60"]),
              length(df$Position[df$rbp == "UPF1" & df$group == "EGF0"]), length(df$Position[df$
    rbp == "UPF1" & df$group == "EGF15"]),
                length(df$Position[df$rbp == "UPF1" & df$group == "EGF30"]), length(df$Position[
    df$rbp == "UPF1" & df$group == "EGF60"]),
```

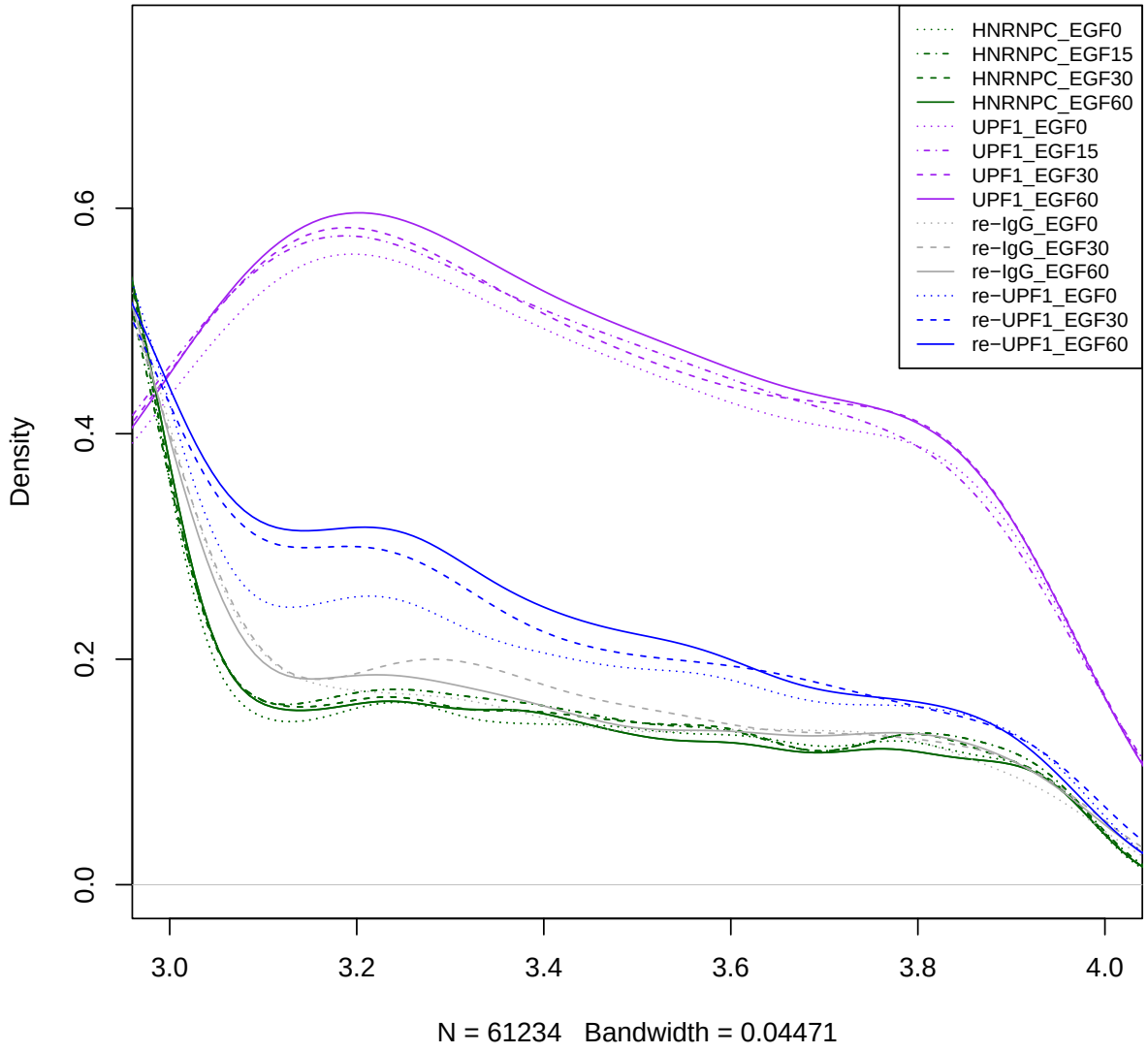
```
            length(df$Position[df$rbp == "re-IgG" & df$group == "EGF0"]), length(df$Position[df
    $rbp == "re-IgG" & df$group == "EGF30"]),
            length(df$Position[df$rbp == "re-IgG" & df$group == "EGF60"]),
            length(df$Position[df$rbp == "re-UPF1" & df$group == "EGF0"]), length(df$Position[
    df$rbp == "re-UPF1" & df$group == "EGF30"]),
            length(df$Position[df$rbp == "re-UPF1" & df$group == "EGF60"])
            ))


D2.hm <- data.frame(HNRNPC_EGF0 = c(df$Position[df$rbp == "HNRNPC" & df$group == "EGF0"],rep(NA,
    length - length(df$Position[df$rbp == "HNRNPC" & df$group == "EGF0"]))),
                    HNRNPC_EGF15 = c(df$Position[df$rbp == "HNRNPC" & df$group == "EGF15"],rep(NA
    , length - length(df$Position[df$rbp == "HNRNPC" & df$group == "EGF15"]))),
                    HNRNPC_EGF30 = c(df$Position[df$rbp == "HNRNPC" & df$group == "EGF30"],rep(NA
    , length - length(df$Position[df$rbp == "HNRNPC" & df$group == "EGF30"]))),
                    HNRNPC_EGF60 = c(df$Position[df$rbp == "HNRNPC" & df$group == "EGF60"],rep(NA
    , length - length(df$Position[df$rbp == "HNRNPC" & df$group == "EGF60"]))),
                    UPF1_EGF0 = c(df$Position[df$rbp == "UPF1" & df$group == "EGF0"],rep(NA,
    length - length(df$Position[df$rbp == "UPF1" & df$group == "EGF0"]))),
                    UPF1_EGF15 = c(df$Position[df$rbp == "UPF1" & df$group == "EGF15"],rep(NA,
    length - length(df$Position[df$rbp == "UPF1" & df$group == "EGF15"]))),
                    UPF1_EGF30 = c(df$Position[df$rbp == "UPF1" & df$group == "EGF30"],rep(NA,
    length - length(df$Position[df$rbp == "UPF1" & df$group == "EGF30"]))),
                    UPF1_EGF60 = c(df$Position[df$rbp == "UPF1" & df$group == "EGF60"],rep(NA,
    length - length(df$Position[df$rbp == "UPF1" & df$group == "EGF60"]))),
                    re_IgG_EGF0 = c(df$Position[df$rbp == "re_IgG" & df$group == "EGF0"],rep(NA,
    length - length(df$Position[df$rbp == "re_IgG" & df$group == "EGF0"]))),
                    re_IgG_EGF30 = c(df$Position[df$rbp == "re_IgG" & df$group == "EGF30"],rep(NA
    , length - length(df$Position[df$rbp == "re_IgG" & df$group == "EGF30"]))),
                    re_IgG_EGF60 = c(df$Position[df$rbp == "re_IgG" & df$group == "EGF60"],rep(NA
    , length - length(df$Position[df$rbp == "re_IgG" & df$group == "EGF60"]))),
                    re_UPF1_EGF0 = c(df$Position[df$rbp == "re_UPF1" & df$group == "EGF0"],rep(NA
    , length - length(df$Position[df$rbp == "re_UPF1" & df$group == "EGF0"]))),
                    re_UPF1_EGF30 = c(df$Position[df$rbp == "re_UPF1" & df$group == "EGF30"],rep(
    NA, length - length(df$Position[df$rbp == "re_UPF1" & df$group == "EGF30"]))),
                    re_UPF1_EGF60 = c(df$Position[df$rbp == "re_UPF1" & df$group == "EGF60"],rep(
    NA, length - length(df$Position[df$rbp == "re_UPF1" & df$group == "EGF60"]))))

ComplexHeatmap::densityHeatmap(D2.hm, col = rev(brewer.pal(30, "Spectral")) , cluster_columns =
    FALSE, ylim = c(0,4), show_quantiles = TRUE, ylab = "Distribution RT stops")
```


Density heatmap of D2.hm
Distribution RT stops
4
3
2
1
0
100%
75%
mean
50%
25%
0%
density
1
0.8
0.6
0.4
0.2
0
HNRNPC_EGF0
HNRNPC_EGF15
HNRNPC_EGF30
HNRNPC_EGF60
UPF1_EGF0
UPF1_EGF15
UPF1_EGF30
UPF1_EGF60
re_IgG_EGF0
re_IgG_EGF30
re_IgG_EGF60
re_UPF1_EGF0
re_UPF1_EGF30
re_UPF1_EGF60

```
# Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/1_Distribution/
    Distribution_density_HNRNPC_UPF1_allregions.pdf", height = 10, width = 4.5)
ComplexHeatmap::densityHeatmap(D2.hm, col = rev(brewer.pal(30, "Spectral")), cluster_columns =
    FALSE, ylim = c(0,4), show_quantiles = TRUE, ylab = "Density of RT stops location across RI
    transcripts")
dev.off()
```

3'UTR magnification

```
#3'UTR only
plot(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF0"]), xlim = c(3, 4), ylim = c(0,
    0.75), col="darkgreen", main="3'UTR density", lty=3)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF15"]), col="darkgreen", lty=4)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF30"]), col="darkgreen", lty=2)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF60"]), col="darkgreen", lty=1)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF0"]), col="purple", lty=3)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF15"]), col="purple", lty=4)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF30"]), col="purple", lty=2)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF60"]), col="purple", lty=1)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF0"]), col="darkgreen", lty=3)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF15"]), col="darkgreen", lty=4)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF30"]), col="darkgreen", lty=2)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF60"]), col="darkgreen", lty=1)
lines(density(df$Position[df$rbp == "re_IgG" & df$group == "EGF0"]), col="darkgrey", lty=3)
lines(density(df$Position[df$rbp == "re_IgG" & df$group == "EGF30"]), col="darkgrey", lty=2)
lines(density(df$Position[df$rbp == "re_IgG" & df$group == "EGF60"]), col="darkgrey", lty=1)
lines(density(df$Position[df$rbp == "re_UPF1" & df$group == "EGF0"]), col="blue", lty=3)
lines(density(df$Position[df$rbp == "re_UPF1" & df$group == "EGF30"]), col="blue", lty=2)
lines(density(df$Position[df$rbp == "re_UPF1" & df$group == "EGF60"]), col="blue", lty=1)
legend(x = "topright", lty = c(3,4,2,1,3,4,2,1,3,2,1,3,2,1), text.font = 1, cex = 0.75,
       col= c(rep("darkgreen", 4), rep("purple", 4), rep("darkgrey", 3),  rep("blue", 3)),text.
    col = "black",
       legend=c("HNRNPC_EGF0", "HNRNPC_EGF15", "HNRNPC_EGF30", "HNRNPC_EGF60",
                "UPF1_EGF0", "UPF1_EGF15", "UPF1_EGF30", "UPF1_EGF60",
                "re-IgG_EGF0", "re-IgG_EGF30", "re-IgG_EGF60",
                "re-UPF1_EGF0", "re-UPF1_EGF30", "re-UPF1_EGF60"))
```




```
# Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/1_Distribution/
    Distribution_density_3UTR_HNRNPC_UPF1_allregions.pdf", height = 7.5, width = 7.5)
plot(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF0"]), xlim = c(3, 4), ylim = c(0,
    0.75), col="darkgreen", main="3'UTR density", lty=3)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF15"]), col="darkgreen", lty=4)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF30"]), col="darkgreen", lty=2)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF60"]), col="darkgreen", lty=1)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF0"]), col="purple", lty=3)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF15"]), col="purple", lty=4)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF30"]), col="purple", lty=2)
lines(density(df$Position[df$rbp == "UPF1" & df$group == "EGF60"]), col="purple", lty=1)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF0"]), col="darkgreen", lty=3)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF15"]), col="darkgreen", lty=4)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF30"]), col="darkgreen", lty=2)
lines(density(df$Position[df$rbp == "HNRNPC" & df$group == "EGF60"]), col="darkgreen", lty=1)
lines(density(df$Position[df$rbp == "re_IgG" & df$group == "EGF0"]), col="darkgrey", lty=3)
lines(density(df$Position[df$rbp == "re_IgG" & df$group == "EGF30"]), col="darkgrey", lty=2)
```

```
lines(density(df$Position[df$rbp == "re_IgG" & df$group == "EGF60"]), col="darkgrey", lty=1)
lines(density(df$Position[df$rbp == "re_UPF1" & df$group == "EGF0"]), col="blue", lty=3)
lines(density(df$Position[df$rbp == "re_UPF1" & df$group == "EGF30"]), col="blue", lty=2)
lines(density(df$Position[df$rbp == "re_UPF1" & df$group == "EGF60"]), col="blue", lty=1)
legend(x = "topright", lty = c(3,4,2,1,3,4,2,1,3,2,1,3,2,1), text.font = 1, cex = 0.75,
       col= c(rep("darkgreen", 4), rep("purple", 4), rep("darkgrey", 3),  rep("blue", 3)),text.
    col = "black",
       legend=c("HNRNPC_EGF0", "HNRNPC_EGF15", "HNRNPC_EGF30", "HNRNPC_EGF60",
                "UPF1_EGF0", "UPF1_EGF15", "UPF1_EGF30", "UPF1_EGF60",
                "re-IgG_EGF0", "re-IgG_EGF30", "re-IgG_EGF60",
                "re-UPF1_EGF0", "re-UPF1_EGF30", "re-UPF1_EGF60"))
dev.off()
```

```
#Test irCLIP
ks.test.reclip <- data.frame(reUPF160vsreIgG0 = ks.test(df$Position[df$Position > 3 & df$rbp == "
    re_IgG" & df$group == "EGF0"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" & df$group
    == "EGF60"])$p.value, reUPF160vsreIgG30 = ks.test(df$Position[df$Position > 3 & df$rbp == "re
    _IgG" & df$group == "EGF30"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" & df$group ==
     "EGF60"])$p.value , reUPF160vsreIgG60 = ks.test(df$Position[df$Position > 3 & df$rbp == "re_
    IgG" & df$group == "EGF60"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" & df$group ==
    "EGF60"])$p.value,
                                reUPF160vsreHNRNPC0 = ks.test(df$Position[df$Position > 3 & df$rbp ==
     "HNRNPC" & df$group == "EGF0"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" & df$group
     == "EGF60"])$p.value, reUPF160vsreHNRNPC15 = ks.test(df$Position[df$Position > 3 & df$rbp ==
     "HNRNPC" & df$group == "EGF15"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" & df$
    group == "EGF60"])$p.value, reUPF160vsreHNRNPC30 = ks.test(df$Position[df$Position > 3 & df$
    rbp == "HNRNPC" & df$group == "EGF30"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" &
    df$group == "EGF60"])$p.value , reUPF160vsreHNRNPC60 = ks.test(df$Position[df$Position > 3 &
    df$rbp == "HNRNPC" & df$group == "EGF60"], df$Position[df$Position > 3 & df$rbp == "re_UPF1"
    & df$group == "EGF60"])$p.value,
                           reUPF160vsreUPF10 = ks.test(df$Position[df$Position > 3 & df$rbp == "re_
    UPF1" & df$group == "EGF0"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" & df$group ==
    "EGF60"])$p.value, reUPF160vsreUPF130 = ks.test(df$Position[df$Position > 3 & df$rbp == "re_
    UPF1" & df$group == "EGF30"], df$Position[df$Position > 3 & df$rbp == "re_UPF1" & df$group ==
     "EGF60"])$p.value
)

ks.test.reclip
```

```
##   reUPF160vsreIgG0 reUPF160vsreIgG30 reUPF160vsreIgG60 reUPF160vsreHNRNPC0
## 1     3.339214e-05       0.0005054441       3.202764e-06                  0
##   reUPF160vsreHNRNPC15 reUPF160vsreHNRNPC30 reUPF160vsreHNRNPC60
## 1         1.187939e-14                    0         7.620571e-13
##   reUPF160vsreUPF10 reUPF160vsreUPF130
## 1     3.607423e-05          0.09059375
```

3. Analysis of differentially bound regions across EGF stimulation

Since previous results point to the 3'UTR as the region where the HNRNPC-UPF1 co-binding is happening, we decided to analyze the total signal across these regions during EGF stimulation. We selected 3'UTRs of RI transcripts having maximal Zscore(log2FC) > 0 against time point 0.. We termed these regions: EGF-responsive 3'UTRs of RI transcripts.

```
#Get AS_events
AS_event <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization
    /1_Distribution/PC_RI_events.txt", header = TRUE)
```

```
AS_event <- AS_event %>% mutate("gene_id" = sapply(strsplit(AS_event$region, "_"), function(x) x
    [1]),
                                "gene_name"= sapply(strsplit(AS_event$region, "_"), function(x) x
    [2]),
                                "chr"= sapply(strsplit(AS_event$region, "_"), function(x) x[3]),
                                "strand"= sapply(strsplit(AS_event$region, "_"), function(x) x
    [4]),
                                "start" = sapply(strsplit(AS_event$region, "_"), function(x) x
    [5]),
                                "end"= sapply(strsplit(AS_event$region, "_"), function(x) x[6]))
    %>% dplyr::select(c(region,chr,start,end, strand, gene_id, gene_name, AS))

AS_event_tx <- merge(AS_event %>% dplyr::select(region,gene_id,AS), gtf_df[gtf_df$type == "
    transcript",], by = "gene_id") %>% dplyr::select(c(seqnames,start,end,transcript_id, strand,
    gene_id, gene_name, AS))

AS_event_bed <- AS_event %>% dplyr::select(chr, start, end, strand, region, gene_id, AS)
AS_event_tx_bed <- AS_event_tx %>% dplyr::select(seqnames, start, end, strand, transcript_id,
    gene_id)

#bedtools to get regions inside transcripts
AS_event_tx2 <- bedtoolsr::bt.intersect(AS_event_bed, AS_event_tx_bed, loj = TRUE)
AS_event_tx2 <- unique(AS_event_tx2)

colnames(AS_event_tx2) <- c("AS_chr", "AS_start", "AS_end", "AS_strand", "AS_region", "AS_geneID"
    , "AS_event", "TX_chr", "TX_start", "TX_end", "TX_strand", "TX_id", "TX_geneID")
AS_event_tx2 <- AS_event_tx2 %>% mutate(geneIDmatch = case_when(AS_geneID != TX_geneID ~ "0",AS_
    geneID == TX_geneID ~ "1"))
AS_event_tx2 <- subset(AS_event_tx2, geneIDmatch > 0)

#Load annotation
txdb <- loadDb("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/siRNA_EGF_SG/2_EGF/8_DTE_analysis
    /0_UPF1_HNRNPC/1_DTE/Genome/gencode.v39.annotation.sqlite")
k <- keys(txdb, keytype = "TXNAME")
tx2gene <- AnnotationDbi::select(txdb, k, "GENEID", "TXNAME")
k <- keys(txdb, keytype = "EXONNAME")
ex2gene <- AnnotationDbi::select(txdb, k, "GENEID", "EXONNAME")

k <- keys(txdb, keytype = "EXONNAME")
ex2gene <- AnnotationDbi::select(txdb, k, "GENEID", "EXONNAME")

#3utr
utr3 <- unlist(threeUTRsByTranscript(txdb, use.names=TRUE))
utr3$TXNAME <- names(utr3)
names(utr3) <- NULL
u3 <- annoGR2DF(utr3)
u3 <- merge(u3, tx2gene, by = "TXNAME")
u3 <- u3 %>% dplyr::rename(TX_id = TXNAME)

u3_AS_event <- merge(AS_event_tx2, u3, by = "TX_id")

#Granges function
granges_AS <- function(AS_type, start, end){
  AS <- AS_event
  AS <- data.frame("region" = AS$region,
                   "gene_id" = sapply(strsplit(AS$region, "_"), function(x) x[1]),
                   "gene_name"= sapply(strsplit(AS$region, "_"), function(x) x[2]),
                   "chr"= sapply(strsplit(AS$region, "_"), function(x) x[3]),
                    "start" = sapply(strsplit(AS$region, "_"), function(x) x[start]),
                   "end"= sapply(strsplit(AS$region, "_"), function(x) x[end]),
                   "strand" = sapply(strsplit(AS$region, "_"), function(x) x[4]),
                   "AS" = AS$AS )

  AS_u3.gr <- makeGRangesFromDataFrame(u3_AS_event[u3_AS_event$AS_event == AS_type, ], keep.extra
    .columns = TRUE)
  mcols(AS_u3.gr) <- mcols(AS_u3.gr)[6]
  AS_u3.gr <- unique_granges(AS_u3.gr)
  colnames(mcols(AS_u3.gr))[colnames(mcols(AS_u3.gr)) == "AS_region"] <- "region"
```

```
  AS_u3.gr$AS_event <- AS_type
  return(AS_u3.gr)
}

#Prepare granges
AS_utr3reg.gr <- granges_AS("RI", 7, 10)
saveRDS(AS_utr3reg.gr, "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization
    /2_Heatmap/UTR3_event_granges.rds")
AS_utr3reg.gr$AS_region <- AS_utr3reg.gr$region
AS_utr3reg.gr$region <- sapply(strsplit(AS_utr3reg.gr$region, "_"), function(x) x[2])
AS_utr3reg.gr <- split(AS_utr3reg.gr, AS_utr3reg.gr$region)

for (i in 1:length(names(AS_utr3reg.gr))) {
  AS_utr3reg.gr[[i]] <- reduceWithMcols(AS_utr3reg.gr[[i]])
}

AS_utr3reg.gr <- unlist(AS_utr3reg.gr)
AS_utr3reg.gr$region <- sapply(AS_utr3reg.gr$region, function(x) {paste(unique(unlist(strsplit(x,
    split = "|", fixed = TRUE))), collapse = "|")})
AS_utr3reg.gr$AS_event <- sapply(AS_utr3reg.gr$AS_event, function(x) {paste(unique(unlist(
    strsplit(x, split = "|", fixed = TRUE))), collapse = "|")})
AS_utr3reg.gr$AS_region <- sapply(AS_utr3reg.gr$AS_region, function(x) {paste(unique(unlist(
    strsplit(x, split = "|", fixed = TRUE))), collapse = "|")})
names(AS_utr3reg.gr) <- NULL
AS_utr3reg.gr$ID <- paste(AS_utr3reg.gr$region,"UTR",1:length(AS_utr3reg.gr$region), sep = "_")

#Coverage function
coverage_matrix <- function(gr, rbp, cond){
gr.p <- subset(gr, strand == "+")
gr.p$width_region <- end(gr.p) - start(gr.p)
gr.m <- subset(gr, strand == "-")
gr.m$width_region <- end(gr.m) - start(gr.m)

#Load bigwigfile
dir <- "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/2_Bigwig/"
bw.EGF0.r1.p <- import.bw(paste(dir, rbp, "_A431_EGF0", cond,"R1.p.bw", sep = ""))
bw.EGF0.r1.m <- import.bw(paste(dir, rbp, "_A431_EGF0", cond,"R1.m.bw", sep = ""))
bw.EGF0.r2.p <- import.bw(paste(dir, rbp, "_A431_EGF0", cond,"R2.p.bw", sep = ""))
bw.EGF0.r2.m <- import.bw(paste(dir, rbp, "_A431_EGF0", cond,"R2.m.bw", sep = ""))

bw.EGF15.r1.p <- import.bw(paste(dir, rbp, "_A431_EGF15", cond,"R1.p.bw", sep = ""))
bw.EGF15.r1.m <- import.bw(paste(dir, rbp, "_A431_EGF15", cond,"R1.m.bw", sep = ""))
bw.EGF15.r2.p <- import.bw(paste(dir, rbp, "_A431_EGF15", cond,"R2.p.bw", sep = ""))
bw.EGF15.r2.m <- import.bw(paste(dir, rbp, "_A431_EGF15", cond,"R2.m.bw", sep = ""))

bw.EGF30.r1.p <- import.bw(paste(dir, rbp, "_A431_EGF30", cond,"R1.p.bw", sep = ""))
bw.EGF30.r1.m <- import.bw(paste(dir, rbp, "_A431_EGF30", cond,"R1.m.bw", sep = ""))
bw.EGF30.r2.p <- import.bw(paste(dir, rbp, "_A431_EGF30", cond,"R2.p.bw", sep = ""))
bw.EGF30.r2.m <- import.bw(paste(dir, rbp, "_A431_EGF30", cond,"R2.m.bw", sep = ""))

bw.EGF60.r1.p <- import.bw(paste(dir, rbp, "_A431_EGF60", cond,"R1.p.bw", sep = ""))
bw.EGF60.r1.m <- import.bw(paste(dir, rbp, "_A431_EGF60", cond,"R1.m.bw", sep = ""))
bw.EGF60.r2.p <- import.bw(paste(dir, rbp, "_A431_EGF60", cond,"R2.p.bw", sep = ""))
bw.EGF60.r2.m <- import.bw(paste(dir, rbp, "_A431_EGF60", cond,"R2.m.bw", sep = ""))

#Create profile matrices
gr.p$EGF0_1 <- getCountsByRegions(bw.EGF0.r1.p, gr.p)
gr.p$EGF0_2 <- getCountsByRegions(bw.EGF0.r2.p, gr.p)
gr.p$EGF15_1 <- getCountsByRegions(bw.EGF15.r1.p, gr.p)
gr.p$EGF15_2 <- getCountsByRegions(bw.EGF15.r2.p, gr.p)
gr.p$EGF30_1 <- getCountsByRegions(bw.EGF30.r1.p, gr.p)
gr.p$EGF30_2 <- getCountsByRegions(bw.EGF30.r2.p, gr.p)
gr.p$EGF60_1 <- getCountsByRegions(bw.EGF60.r1.p, gr.p)
gr.p$EGF60_2 <- getCountsByRegions(bw.EGF60.r2.p, gr.p)

gr.m$EGF0_1 <- getCountsByRegions(bw.EGF0.r1.m, gr.m)
gr.m$EGF0_2 <- getCountsByRegions(bw.EGF0.r2.m, gr.m)
gr.m$EGF15_1 <- getCountsByRegions(bw.EGF15.r1.m, gr.m)
```

```
gr.m$EGF15_2 <- getCountsByRegions(bw.EGF15.r2.m, gr.m)
gr.m$EGF30_1 <- getCountsByRegions(bw.EGF30.r1.m, gr.m)
gr.m$EGF30_2 <- getCountsByRegions(bw.EGF30.r2.m, gr.m)
gr.m$EGF60_1 <- getCountsByRegions(bw.EGF60.r1.m, gr.m)
gr.m$EGF60_2 <- getCountsByRegions(bw.EGF60.r2.m, gr.m)

#Merge replicates
gr <- bind_ranges(gr.p, gr.m)

all <- as.data.frame(mcols(gr))
colnames(all)[grep("EGF", colnames(all))] <- paste(rbp, colnames(all)[grep("EGF", colnames(all))
    ], sep = "_")
df

return(all)
}

coverage_matrix_KD <- function(gr, rbp, cond){
gr.p <- subset(gr, strand == "+")
gr.p$width_region <- end(gr.p) - start(gr.p)
gr.m <- subset(gr, strand == "-")
gr.m$width_region <- end(gr.m) - start(gr.m)

#Load bigwigfile
dir <- "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/5_Knockdown/2_Bigwig/"
bw.EGF0Ctrl.r1.p <- import.bw(paste(dir, rbp, "_CTRi_EGF0_R1.p.bw", sep = ""))
bw.EGF0Ctrl.r1.m <- import.bw(paste(dir, rbp, "_CTRi_EGF0_R1.m.bw", sep = ""))
bw.EGF0Ctrl.r2.p <- import.bw(paste(dir, rbp, "_CTRi_EGF0_R2.p.bw", sep = ""))
bw.EGF0Ctrl.r2.m <- import.bw(paste(dir, rbp, "_CTRi_EGF0_R2.m.bw", sep = ""))

bw.EGF0KD.r1.p <- import.bw(paste(dir, rbp, cond, "EGF0_R1.p.bw", sep = ""))
bw.EGF0KD.r1.m <- import.bw(paste(dir, rbp, cond, "EGF0_R1.m.bw", sep = ""))
bw.EGF0KD.r2.p <- import.bw(paste(dir, rbp, cond, "EGF0_R2.p.bw", sep = ""))
bw.EGF0KD.r2.m <- import.bw(paste(dir, rbp, cond, "EGF0_R2.m.bw", sep = ""))

bw.EGF60Ctrl.r1.p <- import.bw(paste(dir, rbp, "_CTRi_EGF60_R1.p.bw", sep = ""))
bw.EGF60Ctrl.r1.m <- import.bw(paste(dir, rbp, "_CTRi_EGF60_R1.m.bw", sep = ""))
bw.EGF60Ctrl.r2.p <- import.bw(paste(dir, rbp, "_CTRi_EGF60_R2.p.bw", sep = ""))
bw.EGF60Ctrl.r2.m <- import.bw(paste(dir, rbp, "_CTRi_EGF60_R2.m.bw", sep = ""))

bw.EGF60KD.r1.p <- import.bw(paste(dir, rbp, cond, "EGF60_R1.p.bw", sep = ""))
bw.EGF60KD.r1.m <- import.bw(paste(dir, rbp, cond, "EGF60_R1.m.bw", sep = ""))
bw.EGF60KD.r2.p <- import.bw(paste(dir, rbp, cond, "EGF60_R2.p.bw", sep = ""))
bw.EGF60KD.r2.m <- import.bw(paste(dir, rbp, cond, "EGF60_R2.m.bw", sep = ""))

#Create profile matrices
gr.p$EGF0Ctrl_1 <- getCountsByRegions(bw.EGF0Ctrl.r1.p, gr.p)
gr.p$EGF0Ctrl_2 <- getCountsByRegions(bw.EGF0Ctrl.r2.p, gr.p)
gr.p$EGF0KD_1 <- getCountsByRegions(bw.EGF0KD.r1.p, gr.p)
gr.p$EGF0KD_2 <- getCountsByRegions(bw.EGF0KD.r2.p, gr.p)
gr.p$EGF60Ctrl_1 <- getCountsByRegions(bw.EGF60Ctrl.r1.p, gr.p)
gr.p$EGF60Ctrl_2 <- getCountsByRegions(bw.EGF60Ctrl.r2.p, gr.p)
gr.p$EGF60KD_1 <- getCountsByRegions(bw.EGF60KD.r1.p, gr.p)
gr.p$EGF60KD_2 <- getCountsByRegions(bw.EGF60KD.r2.p, gr.p)

gr.m$EGF0Ctrl_1 <- getCountsByRegions(bw.EGF0Ctrl.r1.m, gr.m)
gr.m$EGF0Ctrl_2 <- getCountsByRegions(bw.EGF0Ctrl.r2.m, gr.m)
gr.m$EGF0KD_1 <- getCountsByRegions(bw.EGF0KD.r1.m, gr.m)
gr.m$EGF0KD_2 <- getCountsByRegions(bw.EGF0KD.r2.m, gr.m)
gr.m$EGF60Ctrl_1 <- getCountsByRegions(bw.EGF60Ctrl.r1.m, gr.m)
gr.m$EGF60Ctrl_2 <- getCountsByRegions(bw.EGF60Ctrl.r2.m, gr.m)
gr.m$EGF60KD_1 <- getCountsByRegions(bw.EGF60KD.r1.m, gr.m)
gr.m$EGF60KD_2 <- getCountsByRegions(bw.EGF60KD.r2.m, gr.m)

#Merge replicates
gr <- bind_ranges(gr.p, gr.m)

all <- as.data.frame(mcols(gr))
```

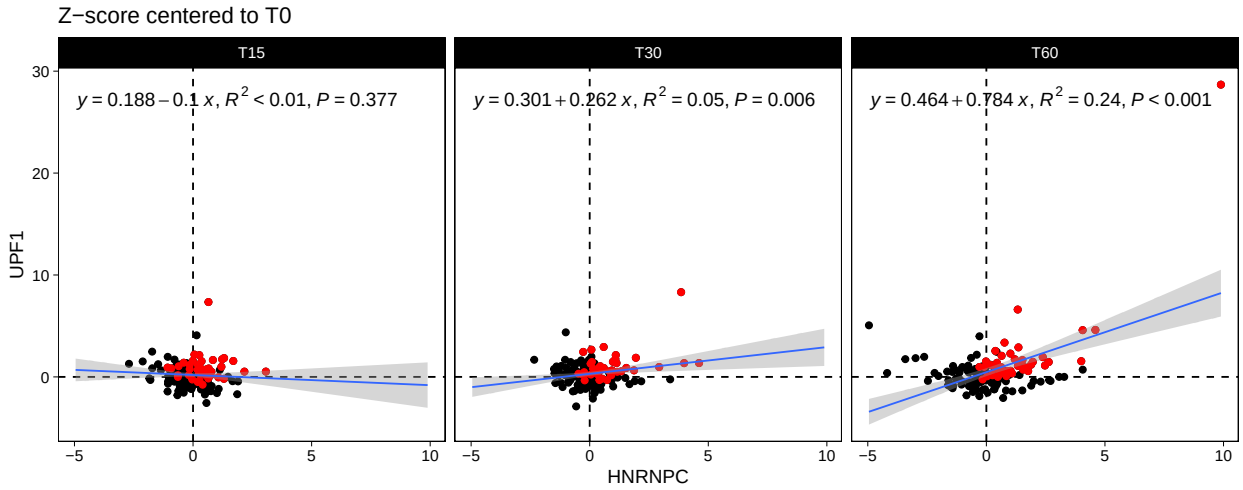
```
colnames(all)[grep("EGF", colnames(all))] <- paste(rbp, colnames(all)[grep("EGF", colnames(all))
    ], sep = "_")
df

return(all)
}

#Timecourse total signal
HNRNPC_3utr <- coverage_matrix(AS_utr3reg.gr, "HNRNPC", "_")
UPF1_3utr <- coverage_matrix(AS_utr3reg.gr, "UPF1", "_HMW_")

#KD total signal
kdHNRNPC_3utr <- coverage_matrix_KD(AS_utr3reg.gr, "HNRNPC", "_UPF1i_")
kdUPF1_3utr <- coverage_matrix_KD(AS_utr3reg.gr, "UPF1", "_hnCi_")

#Save coverage matrix
write.table(HNRNPC_3utr, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_
    Visualization/2_Heatmap/HNRNPC_UTR3_total_signal.txt", row.names = F, quote = F, sep = "\t")
write.table(UPF1_3utr, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_
    Visualization/2_Heatmap/UPF1_UTR3_total_signal.txt", row.names = F, quote = F, sep = "\t")

write.table(kdHNRNPC_3utr, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_
    Visualization/2_Heatmap/kdHNRNPC_UTR3_total_signal.txt", row.names = F, quote = F, sep = "\t"
    )
write.table(kdUPF1_3utr, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_
    Visualization/2_Heatmap/kdUPF1_UTR3_total_signal.txt", row.names = F, quote = F, sep = "\t")
```

Scatterplot of HNRNPC and UPF1 total signal at 3'UTR of RI transcripts during EGF stimulation.

```
#Prepare data for DESeq analysis
HNRNPC_reg <- HNRNPC_3utr
UPF1_reg <- UPF1_3utr

#Timecourse
HNRNPC_design <- data.frame(label = colnames(HNRNPC_reg)[grep("EGF", colnames(HNRNPC_reg))],
    condition = rep(c("T0", "T15", "T30", "T60"), each = 2), replicate = rep(c("1", "2"), 4))
HNRNPC_reg_mat <- as.matrix(HNRNPC_reg[,grep("EGF", colnames(HNRNPC_reg))])
rownames(HNRNPC_reg_mat) <- HNRNPC_reg$ID

UPF1_design <- data.frame(label = colnames(UPF1_reg)[grep("EGF", colnames(UPF1_reg))], condition
    = rep(c("T0", "T15", "T30", "T60"), each = 2), replicate = rep(c("1", "2"), 4))
UPF1_reg_mat <- as.matrix(UPF1_reg[,grep("EGF", colnames(UPF1_reg))])
rownames(UPF1_reg_mat) <- UPF1_reg$ID

#KD
kdHNRNPC_reg <- kdHNRNPC_3utr
kdUPF1_reg <- kdUPF1_3utr

kdHNRNPC_design <- data.frame(label = colnames(kdHNRNPC_reg)[grep("EGF", colnames(kdHNRNPC_reg))
    ], time = rep(c("T0", "T60"), each = 4), replicate = rep(c("1", "2"), 4),
                              condition = rep(c("Ctrl", "KD", "Ctrl", "KD"), each = 2))
kdHNRNPC_reg_mat <- as.matrix(kdHNRNPC_reg[,grep("EGF", colnames(kdHNRNPC_reg))])
rownames(kdHNRNPC_reg_mat) <- kdHNRNPC_reg$ID

kdUPF1_design <- data.frame(label = colnames(kdUPF1_reg)[grep("EGF", colnames(kdUPF1_reg))], time
     = rep(c("T0", "T60"), each = 4), replicate = rep(c("1", "2"), 4),
                              condition = rep(c("Ctrl", "KD", "Ctrl", "KD"), each = 2))
kdUPF1_reg_mat <- as.matrix(kdUPF1_reg[,grep("EGF", colnames(kdUPF1_reg))])
rownames(kdUPF1_reg_mat) <- kdUPF1_reg$ID

#Zscore estimation using DESeq2
deseq_total_rt <- function(data, design, rbp){
```

```
  dds <- DESeqDataSetFromMatrix(data, design, ~ condition)
  keep <- rowSums(counts(dds)) >= 10
  dds <- dds[keep,]
  dds <- estimateSizeFactors(dds)
  normalized_counts <- counts(dds, normalized=TRUE)
  dds <- DESeq(dds)
  res <- results(dds)
  dds_logFC <- coef(dds, SE = F)
  dds_SE <- coef(dds, SE = T)
  dds_cent <- coef(dds, SE = F)/coef(dds, SE = T)
  dds_cent <- dds_cent[,c(2:4)]
  colnames(dds_cent) <- c(paste(rep(paste(rbp, "EGF",sep = "_"), 3), c("15", "30", "60"), sep = "
    _"))
  return(list(table = dds_cent, res = res, norm_counts = normalized_counts, dds = dds))
}

kdHNRNPC_design$condition2 <- paste(kdHNRNPC_design$time, kdHNRNPC_design$condition, sep = "_")
kdUPF1_design$condition2 <- paste(kdUPF1_design$time, kdUPF1_design$condition, sep = "_")

deseq_total_rt_kd <- function(data, design, rbp, TP){
  dds <- DESeqDataSetFromMatrix(data, design, ~ condition)
  keep <- rowSums(counts(dds)) >= 10
  dds <- dds[keep,]
  dds <- estimateSizeFactors(dds)
  dds <- dds[,dds$time == TP]
  normalized_counts <- counts(dds, normalized=TRUE)
  dds <- DESeq(dds)
  res <- results(dds)
  dds_logFC <- coef(dds, SE = F)
  dds_SE <- coef(dds, SE = T)
  dds_cent <- coef(dds, SE = F)/coef(dds, SE = T)
  colnames(dds_cent)[2] <- paste(rbp, TP, "CtrlvsKD", sep = "_")
  return(list(table = dds_cent, res = res, norm_counts = normalized_counts, dds = dds, dds_SE =
    dds_SE))
}

HNRNPC_cent <- deseq_total_rt(HNRNPC_reg_mat, HNRNPC_design, "HNRNPC")
UPF1_cent <- deseq_total_rt(UPF1_reg_mat, UPF1_design, "UPF1")
kdHNRNPC_cent <- deseq_total_rt_kd(kdHNRNPC_reg_mat, kdHNRNPC_design, "HNRNPC", "T0")
kdUPF1_cent <- deseq_total_rt_kd(kdUPF1_reg_mat, kdUPF1_design, "UPF1", "T0")
kdHNRNPC_cent2 <- deseq_total_rt_kd(kdHNRNPC_reg_mat, kdHNRNPC_design, "HNRNPC", "T60")
kdUPF1_cent2 <- deseq_total_rt_kd(kdUPF1_reg_mat, kdUPF1_design, "UPF1", "T60")

merge.all <- function(x, ..., by = "row.names") {
  L <- list(...)
  for (i in seq_along(L)) {
    x <- merge(x, L[[i]], by = by)
    rownames(x) <- x$Row.names
    x$Row.names <- NULL
  }
  return(x)
}

all <- merge.all(HNRNPC_cent$table,UPF1_cent$table, kdHNRNPC_cent$table, kdUPF1_cent$table,
    kdHNRNPC_cent2$table, kdUPF1_cent2$table)

all$HNRNPC_max_lfc <- pmax(all$HNRNPC_EGF_15,all$HNRNPC_EGF_30, all$HNRNPC_EGF_60)
all$HNRNPC_min_lfc <- pmin(all$HNRNPC_EGF_15,all$HNRNPC_EGF_30, all$HNRNPC_EGF_60)
all$HNRNPC_max_lfc_all <- ifelse(abs(all$HNRNPC_max_lfc) > abs(all$HNRNPC_min_lfc), all$HNRNPC_
    max_lfc, all$HNRNPC_min_lfc)

all$UPF1_max_lfc <- pmax(all$UPF1_EGF_15,all$UPF1_EGF_30, all$UPF1_EGF_60)
all$UPF1_min_lfc <- pmin(all$UPF1_EGF_15,all$UPF1_EGF_30, all$UPF1_EGF_60)
all$UPF1_max_lfc_all <- ifelse(abs(all$UPF1_max_lfc) > abs(all$UPF1_min_lfc), all$UPF1_max_lfc,
    all$UPF1_min_lfc)

#Scatterplot
```

```
merge1 <- reshape2::melt(all %>% dplyr::select(HNRNPC_EGF_15:HNRNPC_EGF_60), value.name = "HNRNPC
    ")
merge1$ID <- rep(rownames(all), length(levels(factor(merge1$variable))))
merge2 <- reshape2::melt(all %>% dplyr::select(UPF1_EGF_15:UPF1_EGF_60), value.name = "UPF1")
merge2$ID <- rep(rownames(all), length(levels(factor(merge2$variable))))
merge_melt <- cbind(merge1[,c(1,2)],merge2[,c(2,3)])
merge_melt$TP <- rep(c("T15", "T30", "T60") , each = dim(all)[1])

ggplot3 <- ggplot(data=merge_melt, aes(x=HNRNPC, y=UPF1)) +
  geom_hline(yintercept=0, linetype="dashed", color = "black") +
  geom_vline(xintercept=0, linetype="dashed", color = "black") +
  geom_point() +
  geom_point(data = subset(merge_melt, ID %in% rownames(all)[(all$HNRNPC_max_lfc_all > 0 & all$
    UPF1_max_lfc_all > 0)]), color = "red" ) +
  geom_smooth(method='lm', formula= y~x, size=0.5, fullrange=TRUE) +
  stat_poly_eq(formula = y ~ x,
               aes(label = paste(..eq.label.., ..rr.label.., ..p.value.label.., sep = "*','~")),
               parse = TRUE,
               label.x.npc = "left",
               vstep = 0.05)+
  theme_linedraw() + theme(panel.grid.major = element_blank(), legend.position = "bottom",
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_blank()) + ggtitle("Z-score centered to T0") + facet_wrap(~TP, nrow =
    1)

ggplot3
```

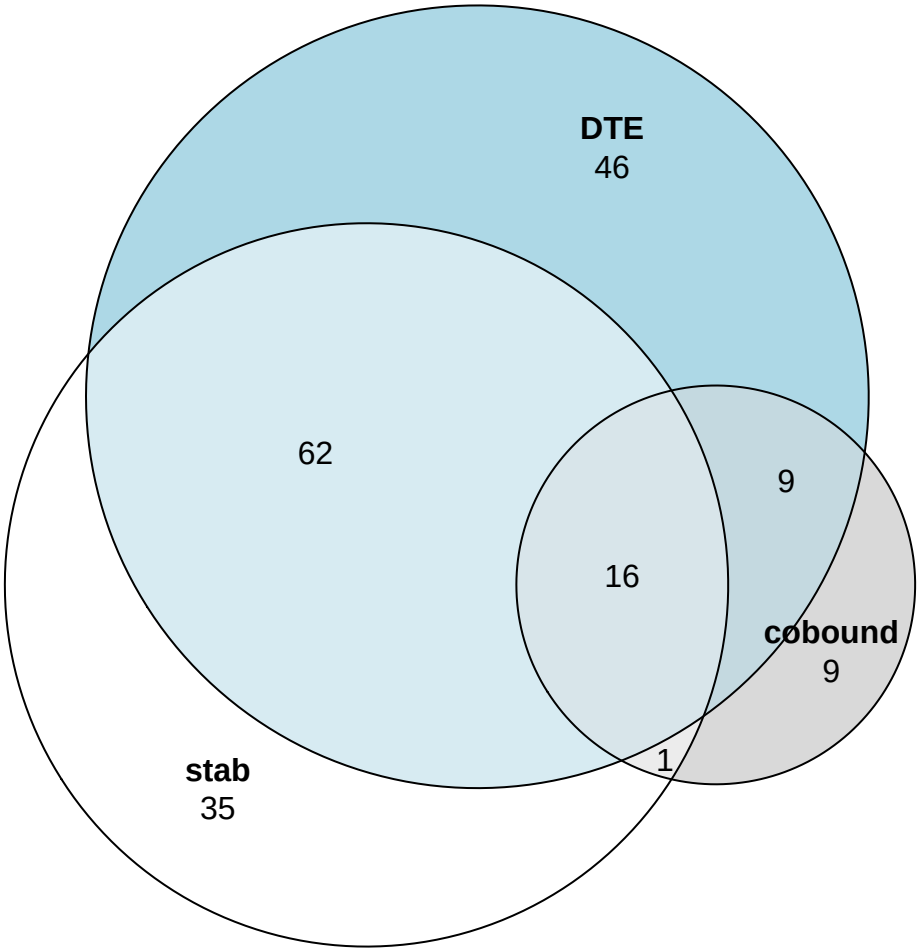



```
#Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/Zscore_
    corr.pdf", height = 4, width = 10)
ggplot3
dev.off()
```

4. Integration with differential expression and stability results

At this point, we integrated the differential expression and stability results with the genes of EGF-responsive RI 3'UTRs to identify which regions are co-bound and co-regulated (co-BR) by HNRNPC and UPF1.

```
#Subset results
all <- subset(all, HNRNPC_max_lfc_all > 0 & UPF1_max_lfc_all > 0)

AS_utr3reg.gr.sub <- AS_utr3reg.gr[AS_utr3reg.gr$ID %in% HNRNPC_reg$ID[HNRNPC_reg$ID %in%
    rownames(all)]]
cobound <- as.data.frame(mcols(AS_utr3reg.gr.sub))

#Overlap between cobound and stability and DE genes
tx2gene <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/siRNA_EGF_SG/2_EGF/4_mRNA_
    stability_assay/4_DTE_timecourse/Analysis/1_DTE/Annotation_transcripts_genes.txt", header =
    TRUE)

#DTE results
DTE <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/siRNA_EGF_SG/2_EGF/8_DTE_
    analysis/0_UPF1_HNRNPC/1_DTE/AS_HNRNPC_UPF1_logFC_high_sign.txt", header = TRUE)
colnames(DTE)[6] <- "GENEID"
DTE <- merge(DTE, tx2gene, by = "GENEID", all.x = TRUE)

#Stability results
stab <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/siRNA_EGF_SG/2_EGF/4_mRNA_
    stability_assay/4_DTE_timecourse/Analysis/1_DTE/Limma_sign_transcript_upset.txt", header =
    TRUE)
colnames(stab)[1] <- "TXNAME"
stab <- merge(stab, tx2gene, by = "TXNAME", all.x = TRUE)

lt.tsk = list(stab = unique(stab$SYMBOL),
              cobound = unique(cobound$region),
              DTE = unique(DTE$SYMBOL)
)

fromList <- function (input) {
  elements <- unique(unlist(input))
  data <- unlist(lapply(input, function(x) {
    x <- as.vector(match(elements, x))
  }))
  data[is.na(data)] <- as.integer(0)
  data[data != 0] <- as.integer(1)
  data <- data.frame(matrix(data, ncol = length(input), byrow = F))
  data <- data[which(rowSums(data) != 0), ]
  names(data) <- names(input)
  # ... Except now it conserves your original value names!
  row.names(data) <- elements
  return(data)
}

# Binary table with colnames:
sign.proteins2 <- fromList(lt.tsk)
sign.proteins2$funct <- paste(sign.proteins2$stab, sign.proteins2$cobound, sign.proteins2$DTE,
    sep = "_")

plot(euler(lt.tsk), quantities = TRUE)
```



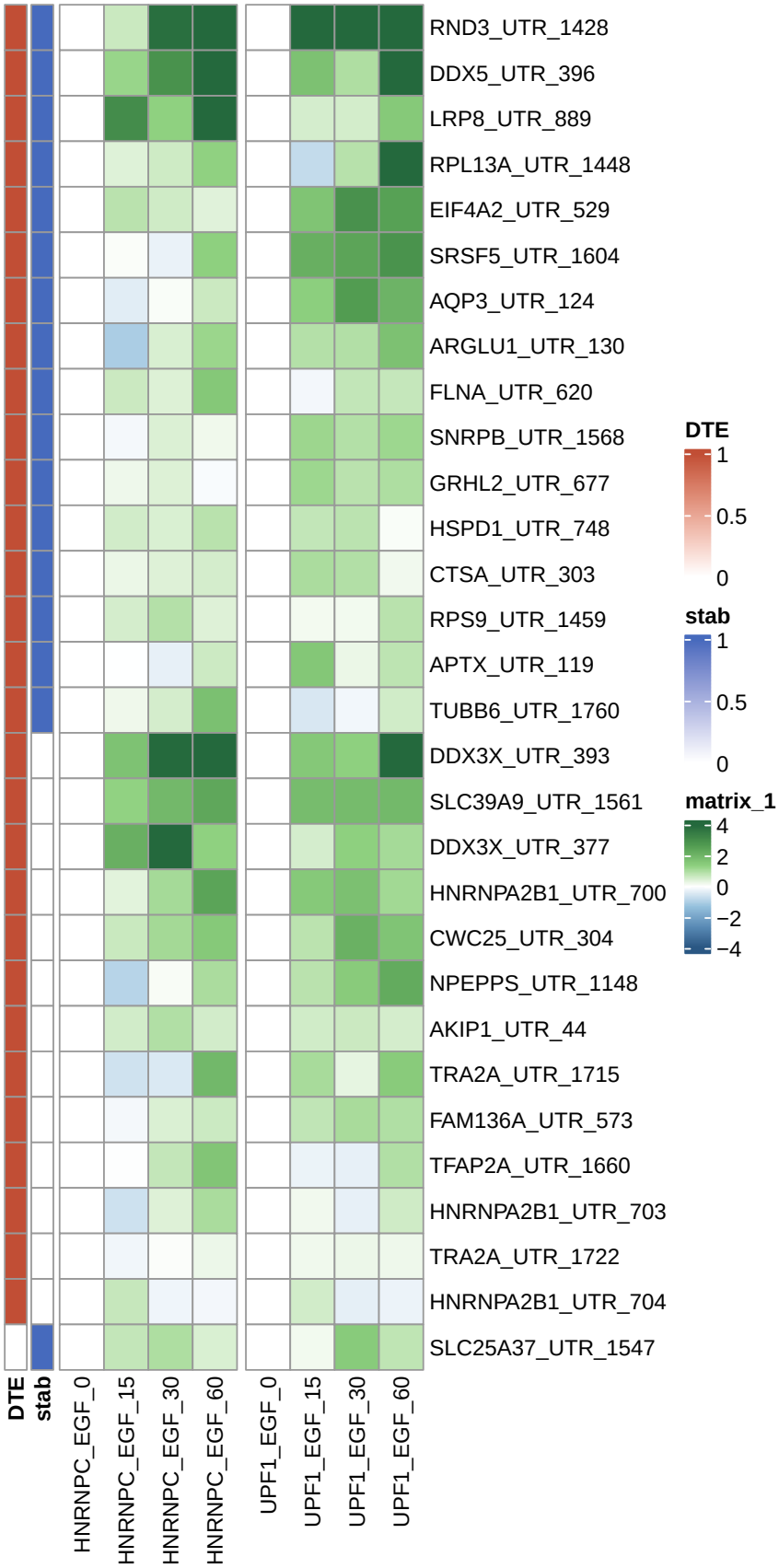

```
cobound.funct <- cobound[cobound$region %in% rownames(sign.proteins2)[sign.proteins2$funct %in% c
    ("0_1_1", "1_1_0", "1_1_1")],]

cobound_exp <- as.data.frame(cobound.funct %>% separate_rows(AS_region, sep = "\\|"))
rMATs.tr <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/siRNA_EGF_SG/2_EGF/8_DTE_
    analysis/0_UPF1_HNRNPC/1_DTE/AS_HNRNPC_UPF1_transcripts.txt", header = TRUE)

cobound_exp <- merge(cobound_exp, rMATs.tr %>% dplyr::select(AS_region, TX_id, TX_geneID), by = "
    AS_region")
cobound_exp$stab <- ifelse(cobound_exp$TX_id %in% stab$TXNAME, 1, 0)
cobound_exp$DTE <- ifelse(cobound_exp$TX_id %in% DTE$TX_id, 1, 0)

cobound_funct2 <- unique(cobound_exp %>% dplyr::select(AS_region, region, ID, stab, DTE))

cobound_funct2 <- as.data.frame(cobound_funct2 %>% group_by(ID) %>% summarise(stab = sum(stab, na
    .rm = TRUE), DTE = sum(DTE, na.rm = TRUE)))
cobound_funct2$stab <- ifelse(cobound_funct2$stab > 0, 1, 0)
cobound_funct2$DTE <- ifelse(cobound_funct2$DTE > 0, 1, 0)

AS_utr3reg.gr.sub <- AS_utr3reg.gr[AS_utr3reg.gr$ID %in% cobound.funct$ID]
write.table(as.data.frame(mcols(AS_utr3reg.gr.sub)), file = "~/Documents/Postdoc/PD_Projects/3_
    irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/HNRNPC_UPF1_cobound_genes.txt", quote = F,
    row.names = F, sep = "\t")
```

```
#Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/Overlap_
    DTE_stab_cobound.pdf", height = 5, width = 5)
```

```
plot(euler(lt.tsk), quantities = TRUE)
dev.off()
```

Heatmap of co-bound and co-regulated 3'UTR of RI transcripts in irCLIPv2 datasets

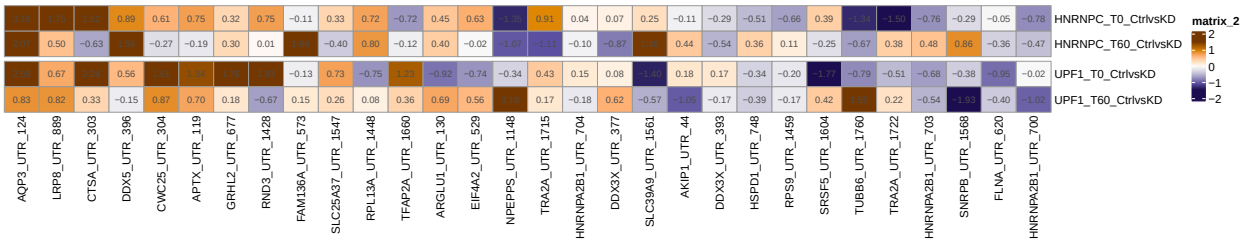
```
#Heatmap timecourse table
hm_table <- all %>% dplyr::select(HNRNPC_EGF_15:UPF1_EGF_60)
hm_table <- hm_table[rownames(hm_table) %in% cobound.funct$ID,]

hm_table2 <- cbind(hm_table, HNRNPC_EGF_0 = rep(0, dim(hm_table)[1]), UPF1_EGF_0 = rep(0, dim(hm_
    table)[1]))
hm_table2 <- hm_table2 %>% dplyr::select(HNRNPC_EGF_0, HNRNPC_EGF_15, HNRNPC_EGF_30, HNRNPC_EGF_
    60,
                                         UPF1_EGF_0, UPF1_EGF_15, UPF1_EGF_30, UPF1_EGF_60)
hm_table2 <- as.matrix(hm_table2)

my.breaks <- c(seq(-4, 4, by=0.01))
my.colors <- rev(c(paletteer_c("ggthemes::Green-Blue-White Diverging", length(my.breaks))))

annotation_row <- cobound_funct2
rownames(annotation_row) <- annotation_row$ID
annotation_row$ID <- NULL
annotation_row <- annotation_row[match(rownames(hm_table2), rownames(annotation_row)), ]
annotation_row$row_sums <- rowSums(hm_table2)
annotation_row <- annotation_row %>% arrange(-row_sums) %>% arrange(-stab) %>% arrange(-DTE)
hm_table2 <- hm_table2[match(rownames(annotation_row), rownames(hm_table2)), ]

ph <- pheatmap(hm_table2, scale = "none", annotation_row = annotation_row[,-3], show_rownames = T
    , cluster_rows = FALSE, clustering_distance_rows = "correlation", cluster_cols = FALSE,
    clustering_method = "ward.D2",
        gaps_col = c(4,8), color = my.colors, breaks = my.breaks, display_numbers = FALSE)#,
    cutree_rows = 4)
ph
```


RND3_UTR_1428
DDX5_UTR_396
LRP8_UTR_889
RPL13A_UTR_1448
EIF4A2_UTR_529
SRSF5_UTR_1604
AQP3_UTR_124
ARGLU1_UTR_130
FLNA_UTR_620
SNRPB_UTR_1568
GRHL2_UTR_677
HSPD1_UTR_748
CTSA_UTR_303
RPS9_UTR_1459
APTX_UTR_119
TUBB6_UTR_1760
DDX3X_UTR_393
SLC39A9_UTR_1561
DDX3X_UTR_377
HNRNPA2B1_UTR_700
CWC25_UTR_304
NPEPPS_UTR_1148
AKIP1_UTR_44
TRA2A_UTR_1715
FAM136A_UTR_573
TFAP2A_UTR_1660
HNRNPA2B1_UTR_703
TRA2A_UTR_1722
HNRNPA2B1_UTR_704
SLC25A37_UTR_1547
DTE
1
0.5
0
stab
1
0.5
0
matrix_1
4
2
0
−2
−4
DTE
stab
HNRNPC_EGF_0
HNRNPC_EGF_15
HNRNPC_EGF_30
HNRNPC_EGF_60
UPF1_EGF_0
UPF1_EGF_15
UPF1_EGF_30
UPF1_EGF_60

```
#Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/Zscore_
    heatmap.pdf", height = 10, width = 5)
ph
dev.off()
```

Heatmap of co-bound and co-regulated 3'UTR of RI transcripts in irCLIPv2 knockdown datasets

```
#Heatmap knockdown
hm_table3 <- all %>% dplyr::select(HNRNPC_T0_CtrlvsKD, HNRNPC_T60_CtrlvsKD, UPF1_T0_CtrlvsKD,
    UPF1_T60_CtrlvsKD)
hm_table3 <- hm_table3[rownames(hm_table3) %in% cobound.funct$ID,]

hm_table3 <- as.matrix(hm_table3)
row_sums2 <- rowSums(hm_table3)
hm_table3 <- hm_table3[order(row_sums2, decreasing = TRUE), ]

my.breaks2 <- c(seq(-1.5, 1.5, by=0.01))
my.colors2 <- rev(c(paletteer_c("grDevices::PuOr", length(my.breaks))))

ph2 <- pheatmap(t(hm_table3), scale = "none", show_rownames = T, cluster_rows = FALSE, cluster_
    cols = FALSE, clustering_method = "ward.D", gaps_row = 2,
        color = my.colors2, breaks = my.breaks2, display_numbers = T)#, cutree_rows = 4)
ph2
```



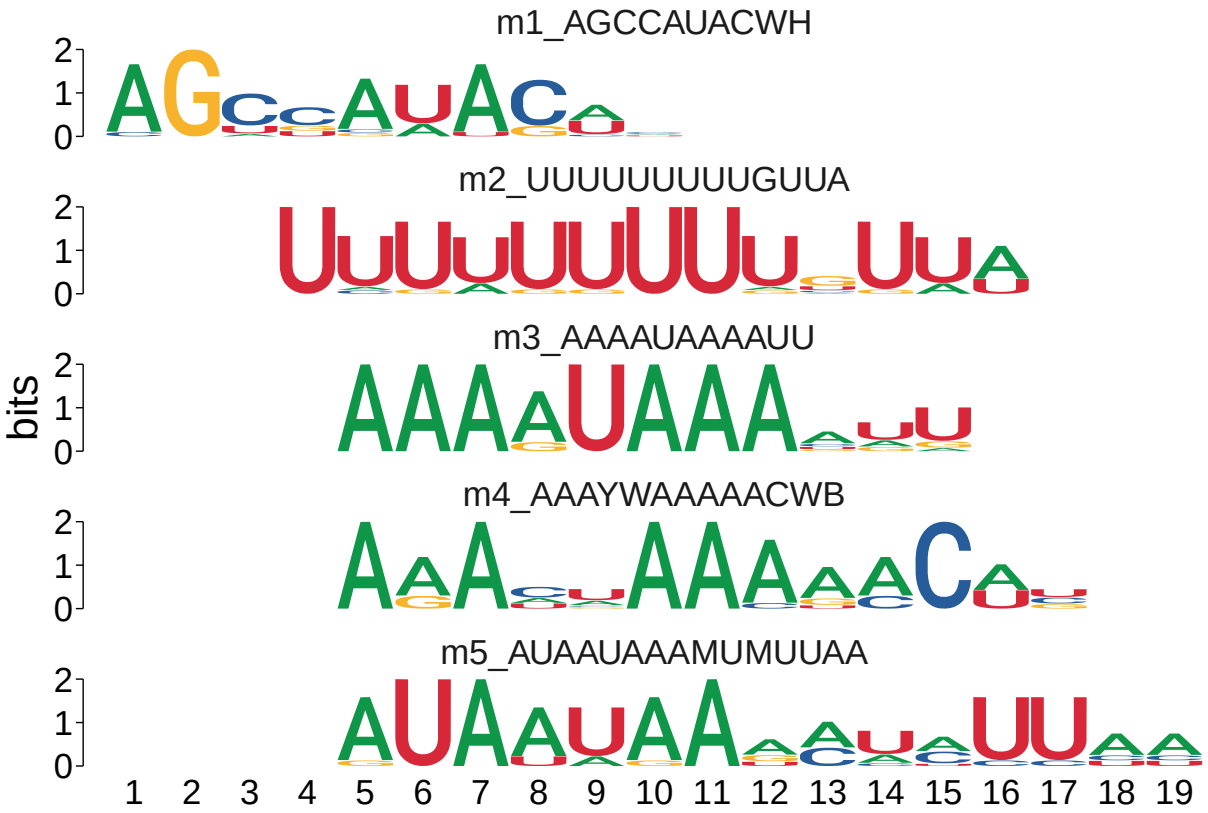

```
#Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/Zscore_
    heatmap_KD.pdf", height = 3, width = 15)
ph2
dev.off()
```

5. De-novo motif analysis using STREME

We have performed de-novo motif analysis of EGF-responsive co-BR 3'UTRs of RI transcipts.

```
#Subset for re-CLIP
AS_utr3reg.gr.sub <- AS_utr3reg.gr.sub[AS_utr3reg.gr.sub$ID %in% cobound.funct$ID,]

#Get sequence
gr.seq <- AS_utr3reg.gr.sub %>% get_sequence(BSgenome.Hsapiens.UCSC.hg38)

streme <- runStreme(gr.seq, control = "shuffle", alph = "rna", minw = 10, maxw=15, align = "
    center")
```

```
streme %>%
  to_list() %>%
  view_motifs(tryRC = FALSE)
```




```
#Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/Motif_
    enrichment.pdf", height = 5, width = 5)
streme %>%
  to_list() %>%
  view_motifs(tryRC = FALSE)
dev.off()
```

5. Analysis of co-BR regions in Re-CLIP dataset

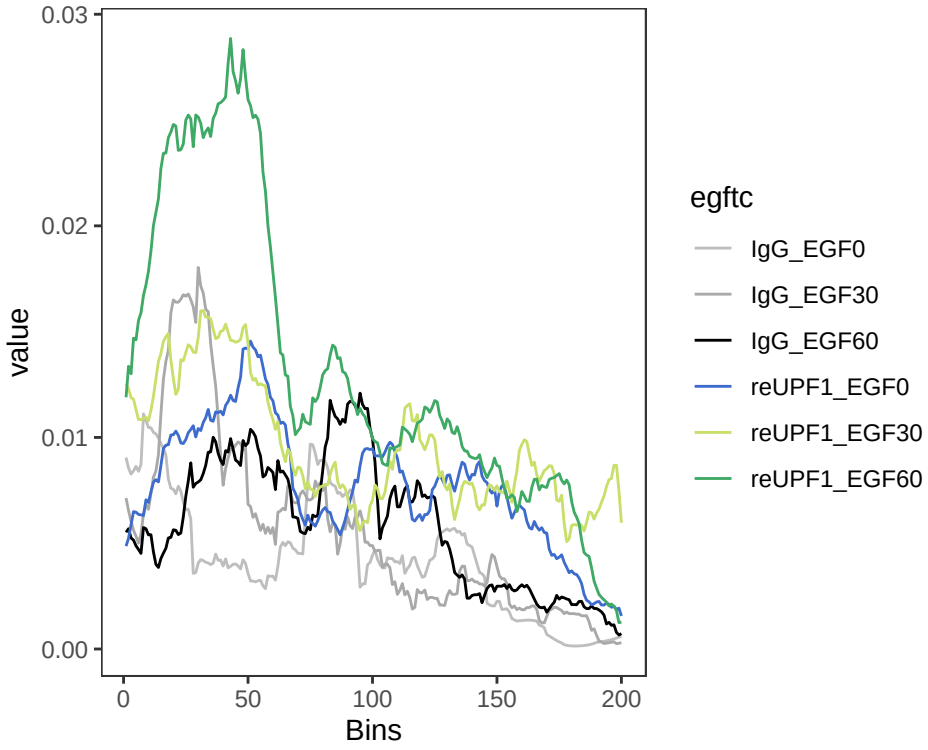
Profile plot of EGF-responsive co-BR 3'UTRs of RI transcripts.

```
#Coverage function
coverage_matrix_rclip <- function(gr, rbp, cond){
  gr.p <- subset(gr, strand == "+")
  gr.p$width_region <- end(gr.p) - start(gr.p)
  gr.m <- subset(gr, strand == "-")
  gr.m$width_region <- end(gr.m) - start(gr.m)

  #Load bigwigfile
```

```
dir <- "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/6_reclip-EGF_tcupf1/3_Bigwig_sum/"
bw.EGF0.r1.p <- import.bw(paste(dir, rbp, "_A431_hnC_EGF0", cond,"R1.p.scaleddewseq.bw", sep = ""))
bw.EGF0.r1.m <- import.bw(paste(dir, rbp, "_A431_hnC_EGF0", cond,"R1.m.scaleddewseq.bw", sep = ""))
bw.EGF0.r2.p <- import.bw(paste(dir, rbp, "_A431_hnC_EGF0", cond,"R2.p.scaleddewseq.bw", sep = ""))
bw.EGF0.r2.m <- import.bw(paste(dir, rbp, "_A431_hnC_EGF0", cond,"R2.m.scaleddewseq.bw", sep = ""))

bw.EGF30.r1.p <- import.bw(paste(dir, rbp, "_A431_hnC_EGF30", cond,"R1.p.scaleddewseq.bw", sep = ""))
bw.EGF30.r1.m <- import.bw(paste(dir, rbp, "_A431_hnC_EGF30", cond,"R1.m.scaleddewseq.bw", sep = ""))
bw.EGF30.r2.p <- import.bw(paste(dir, rbp, "_A431_hnC_EGF30", cond,"R2.p.scaleddewseq.bw", sep = ""))
bw.EGF30.r2.m <- import.bw(paste(dir, rbp, "_A431_hnC_EGF30", cond,"R2.m.scaleddewseq.bw", sep = ""))

bw.EGF60.r1.p <- import.bw(paste(dir, rbp, "_A431_hnC_EGF60", cond,"R1.p.scaleddewseq.bw", sep = ""))
bw.EGF60.r1.m <- import.bw(paste(dir, rbp, "_A431_hnC_EGF60", cond,"R1.m.scaleddewseq.bw", sep = ""))
bw.EGF60.r2.p <- import.bw(paste(dir, rbp, "_A431_hnC_EGF60", cond,"R2.p.scaleddewseq.bw", sep = ""))
bw.EGF60.r2.m <- import.bw(paste(dir, rbp, "_A431_hnC_EGF60", cond,"R2.m.scaleddewseq.bw", sep = ""))

#Create profile matrices#
EGF0.mat1.p = normalizeToMatrix(bw.EGF0.r1.p, gr.p, value_column = "score", mean_mode = "w0",
  extend = 0, k = 200, w = 20, background = 0, smooth = TRUE, target_ratio = 1, limit = NA)
EGF0.mat1.m = normalizeToMatrix(bw.EGF0.r1.m, gr.m, value_column = "score", mean_mode = "w0",
  extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF0.mat2.p = normalizeToMatrix(bw.EGF0.r2.p, gr.p, value_column = "score", mean_mode = "w0",
  extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF0.mat2.m = normalizeToMatrix(bw.EGF0.r2.m, gr.m, value_column = "score", mean_mode = "w0",
  extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)

EGF30.mat1.p = normalizeToMatrix(bw.EGF30.r1.p, gr.p, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF30.mat1.m = normalizeToMatrix(bw.EGF30.r1.m, gr.m, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF30.mat2.p = normalizeToMatrix(bw.EGF30.r2.p, gr.p, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF30.mat2.m = normalizeToMatrix(bw.EGF30.r2.m, gr.m, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)

EGF60.mat1.p = normalizeToMatrix(bw.EGF60.r1.p, gr.p, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF60.mat1.m = normalizeToMatrix(bw.EGF60.r1.m, gr.m, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF60.mat2.p = normalizeToMatrix(bw.EGF60.r2.p, gr.p, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)
EGF60.mat2.m = normalizeToMatrix(bw.EGF60.r2.m, gr.m, value_column = "score", mean_mode = "w0",
   extend = 0, k = 200, w = 20, background = 0, smooth = FALSE, target_ratio = 1, limit = NA)

#Merge replicates
EGF0.mat_list.p = list(EGF0.mat1.p, EGF0.mat2.p)
EGF0.mat.p = getSignalsFromList(EGF0.mat_list.p)
rownames(EGF0.mat.p) <- gr.p$ID

EGF30.mat_list.p = list(EGF30.mat1.p, EGF30.mat2.p)
EGF30.mat.p = getSignalsFromList(EGF30.mat_list.p)
rownames(EGF30.mat.p) <- gr.p$ID

EGF60.mat_list.p = list(EGF60.mat1.p, EGF60.mat2.p)
EGF60.mat.p = getSignalsFromList(EGF60.mat_list.p)
rownames(EGF60.mat.p) <- gr.p$ID
```

```
  EGF0.mat_list.m = list(EGF0.mat1.m, EGF0.mat2.m)
  EGF0.mat.m = getSignalsFromList(EGF0.mat_list.m)
  rownames(EGF0.mat.m) <- gr.m$ID

  EGF30.mat_list.m = list(EGF30.mat1.m, EGF30.mat2.m)
  EGF30.mat.m = getSignalsFromList(EGF30.mat_list.m)
  rownames(EGF30.mat.m) <- gr.m$ID

  EGF60.mat_list.m = list(EGF60.mat1.m, EGF60.mat2.m)
  EGF60.mat.m = getSignalsFromList(EGF60.mat_list.m)
  rownames(EGF60.mat.m) <- gr.m$ID

  #Combine p and m
  EGF0.mat <- rbind(EGF0.mat.p, EGF0.mat.m)
  rownames(EGF0.mat) <- c(gr.p$ID, gr.m$ID)
  EGF30.mat <- rbind(EGF30.mat.p, EGF30.mat.m)
  rownames(EGF30.mat) <- c(gr.p$ID, gr.m$ID)
  EGF60.mat <- rbind(EGF60.mat.p, EGF60.mat.m)
  rownames(EGF60.mat) <- c(gr.p$ID, gr.m$ID)
  return(list(EGF0 = EGF0.mat, EGF30 = EGF30.mat, EGF60 = EGF60.mat))
}

IgG_mat <- coverage_matrix_rclip(AS_utr3reg.gr.sub, "IgG", "_")
UPF1_mat <- coverage_matrix_rclip(AS_utr3reg.gr.sub, "UPF1", "_HMW_")

profileplot <- function(x){
    S1.mat <- x$EGF0[]
    S2.mat <- x$EGF30[]
    S3.mat <- x$EGF60[]

    k = 20
    S1.mat2 = t(runmean(t(S1.mat), k))
    S2.mat2 = t(runmean(t(S2.mat), k))
    S3.mat2 = t(runmean(t(S3.mat), k))

    S1.gg <- data.frame("value" = colMeans(S1.mat2, na.rm=TRUE))
    S1.gg$type <- "EGF0"
    S1.gg$xlab <- c(1:dim(S1.mat2)[2])
    S2.gg <- data.frame("value" = colMeans(S2.mat2, na.rm=TRUE))
    S2.gg$type <- "EGF30"
    S2.gg$xlab <- c(1:dim(S2.mat2)[2])
    S3.gg <- data.frame("value" = colMeans(S3.mat2, na.rm=TRUE))
    S3.gg$type <- "EGF60"
    S3.gg$xlab <- c(1:dim(S3.mat2)[2])

    all <- rbind(S1.gg,S2.gg,S3.gg)
    all$type <- factor(all$type)
    all$type <- factor(all$type, levels = c("EGF60", "EGF30", "EGF0"))
    return(list(all = all, EGF0 = S1.mat2, EGF30 = S2.mat2, EGF60 = S3.mat2))
  }

IgG.plot <- profileplot(IgG_mat)
UPF1.plot <- profileplot(UPF1_mat)

IgG.plot$all$rbp <- "IgG"
UPF1.plot$all$rbp <- "reUPF1"

all_profile <- rbind(IgG.plot$all, UPF1.plot$all)
all_profile$egftc <- paste(all_profile$rbp, all_profile$type, sep = "_")

all.plot <- all_profile %>%
  ggplot( aes(x=xlab, y=value, group = egftc, color = egftc)) +
  geom_line() +
  # scale_linetype_manual(values = c("EGF60" = 1, "EGF30" = 4, "EGF0" = 2)) +
  scale_color_manual(values = c("IgG_EGF0" = "grey", "IgG_EGF30" = "darkgrey", "IgG_EGF60" = "
    black", "reUPF1_EGF60" = "#3fa966", "reUPF1_EGF30" = "#c8e06a", "reUPF1_EGF0"= "#3d6ccd"))
    +
```

```
  xlab("Bins") +
  theme_bw() +
  theme( panel.grid.major = element_blank(),
         panel.grid.minor = element_blank(),
         panel.background = element_blank(),
         axis.line = element_blank(),
         plot.title = element_text(hjust = 0.5)) #+
  #facet_wrap(vars(rbp), ncol = 1, nrow = 2)

all.plot
```

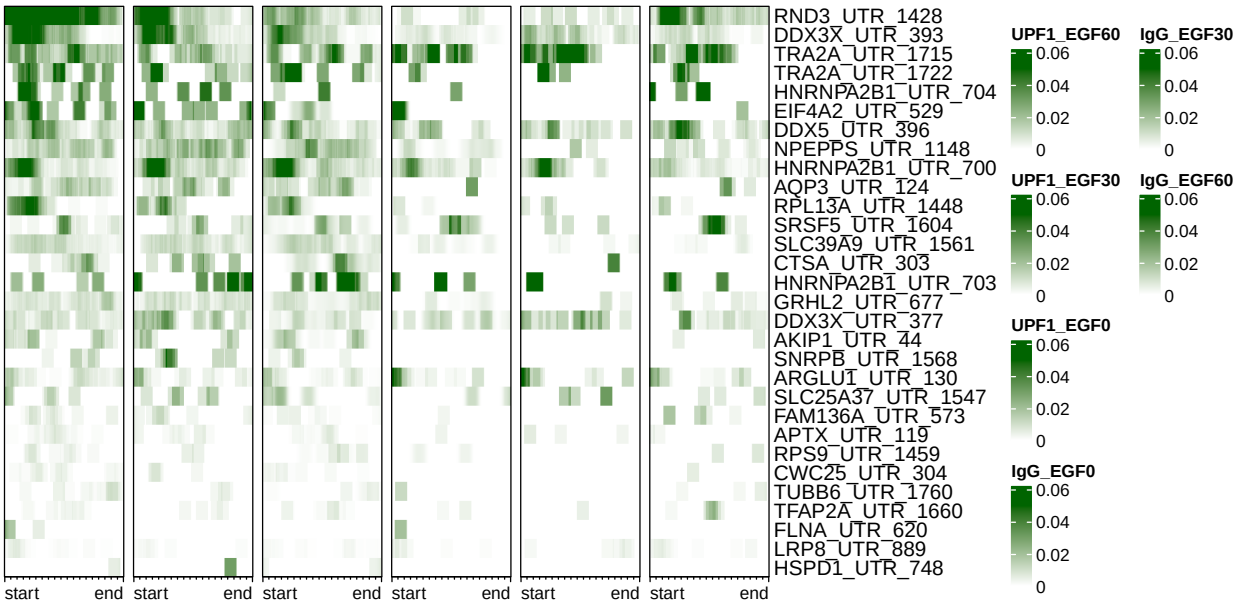



```
#Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/
    Profileplot_reCLIP.pdf", height = 4, width = 5)
all.plot
dev.off()
```

Enriched of EGF-responsive co-BR 3'UTRs of RI transcripts.

```
#Prepare matrices for enriched heatmap
UPF1_mat$EGF60[] <- UPF1.plot$EGF60
UPF1_mat$EGF30[] <- UPF1.plot$EGF30
UPF1_mat$EGF0[] <- UPF1.plot$EGF0

IgG_mat$EGF60[] <- IgG.plot$EGF60
IgG_mat$EGF30[] <- IgG.plot$EGF30
IgG_mat$EGF0[] <- IgG.plot$EGF0

col_fun = colorRamp2(c(0, 0.05), c("white", "darkgreen"))
```

```
#Generate heatmap
EH <- EnrichedHeatmap(UPF1_mat$EGF60, name = "UPF1_EGF60", col = col_fun, top_annotation = NULL
    , use_raster=TRUE, raster_quality = 10) +
  EnrichedHeatmap(UPF1_mat$EGF30, name = "UPF1_EGF30", col = col_fun, top_annotation = NULL, use
    _raster=TRUE, raster_quality = 10) +
    EnrichedHeatmap(UPF1_mat$EGF0, name = "UPF1_EGF0", col = col_fun, top_annotation = NULL, use
    _raster=TRUE, raster_quality = 10) +
  EnrichedHeatmap(IgG_mat$EGF0, name = "IgG_EGF0", col = col_fun, top_annotation = NULL, use_
    raster=TRUE, raster_quality = 10) +
  EnrichedHeatmap(IgG_mat$EGF30, name = "IgG_EGF30", col = col_fun, top_annotation = NULL, use_
    raster=TRUE, raster_quality = 10) +
  EnrichedHeatmap(IgG_mat$EGF60, name = "IgG_EGF60", col = col_fun, show_row_names = TRUE, use_
    raster=TRUE, top_annotation = NULL, raster_quality = 10)

EH
```




```
#Save the plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/Seq/4_EGF_tc/4_Visualization/2_Heatmap/
    EnrichedHeatmap_reCLIP.pdf", height = 5, width = 10)
EH
dev.off()
```

All the visualizations were saved as pdf and modified in illustrator.

```
sessionInfo()
```

```
## R version 4.2.1 (2022-06-23)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
##
## locale:
```

```
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] GenomicFeatures_1.48.4            SLIMFinderR_0.1.1
##  [3] ProtDomSeq_0.1.0                  NetFeaturePval_0.1.0
##  [5] remotes_2.5.0                     universalmotif_1.14.1
##  [7] BSgenome.Hsapiens.UCSC.hg38_1.4.4 BSgenome_1.64.0
##  [9] Biostrings_2.66.0                 XVector_0.38.0
## [11] memes_1.4.1                       eulerr_7.0.1
## [13] ggpmisc_0.5.5                     ggpp_0.5.6
## [15] DESeq2_1.38.3                     SummarizedExperiment_1.28.0
## [17] MatrixGenerics_1.10.0             matrixStats_1.2.0
## [19] BRGenomics_1.8.0                  data.table_1.15.2
## [21] ChIPpeakAnno_3.30.1               RColorBrewer_1.1-3
## [23] NbClust_3.0.1                     gridExtra_2.3
## [25] paletteer_1.6.0                   cliProfiler_1.2.0
## [27] caTools_1.18.2                    circlize_0.4.16
## [29] EnrichedHeatmap_1.26.0            ComplexHeatmap_2.14.0
## [31] rtracklayer_1.56.1                gintools_0.1.3
## [33] plyranges_1.16.0                  GenomicRanges_1.50.2
## [35] GenomeInfoDb_1.34.9               AnnotationDbi_1.60.2
## [37] IRanges_2.32.0                    S4Vectors_0.36.2
## [39] Biobase_2.58.0                    AnnotationHub_3.4.0
## [41] BiocFileCache_2.4.0               dbplyr_2.4.0
## [43] Repitools_1.42.0                  BiocGenerics_0.44.0
## [45] lubridate_1.9.3                   forcats_1.0.0
## [47] stringr_1.5.1                     dplyr_1.1.4
## [49] purrr_1.0.2                       readr_2.1.5
## [51] tidyr_1.3.1                       tibble_3.2.1
## [53] ggplot2_3.5.0                     tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##   [1] rappdirs_0.3.3                SparseM_1.81
##   [3] ggthemes_5.1.0                GGally_2.2.1
##   [5] R.methodsS3_1.8.2             bit64_4.0.5
##   [7] knitr_1.45                    R.utils_2.12.3
##   [9] DelayedArray_0.24.0           KEGGREST_1.38.0
##  [11] RCurl_1.98-1.14               AnnotationFilter_1.22.0
##  [13] doParallel_1.0.17             generics_0.1.3
##  [15] preprocessCore_1.60.2         gsmoothr_0.1.7
##  [17] lambda.r_1.2.4                RSQLite_2.3.5
##  [19] bit_4.0.5                     tzdb_0.4.0
##  [21] xml2_1.3.6                    httpuv_1.6.14
##  [23] xfun_0.42                     hms_1.1.3
##  [25] evaluate_0.23                 DNAcopy_1.70.0
##  [27] promises_1.2.1                fansi_1.0.6
##  [29] restfulr_0.0.15               progress_1.2.3
##  [31] DBI_1.2.2                     geneplotter_1.76.0
##  [33] Rsolnp_1.16                   htmlwidgets_1.6.4
##  [35] futile.logger_1.4.3           ellipsis_0.3.2
##  [37] annotate_1.76.0               prismatic_1.1.1
##  [39] biomaRt_2.52.0                vctrs_0.6.5
##  [41] quantreg_5.97                 ensembldb_2.20.2
##  [43] Cairo_1.6-2                   ROCR_1.0-11
##  [45] cachem_1.0.8                  withr_3.0.0
##  [47] GenomicAlignments_1.34.1      prettyunits_1.2.0
##  [49] cluster_2.1.6                 lazyeval_0.2.2
##  [51] crayon_1.5.2                  genefilter_1.78.0
##  [53] edgeR_3.40.2                  pkgconfig_2.0.3
##  [55] labeling_0.4.3                nlme_3.1-164
##  [57] pkgload_1.3.4                 ProtGenerics_1.30.0
##  [59] rlang_1.1.3                   Ringo_1.60.0
##  [61] lifecycle_1.0.4               MatrixModels_0.5-3
##  [63] filelock_1.0.3                affyio_1.68.0
```

```
##  [65] VennDiagram_1.7.3           rprojroot_2.0.4
##  [67] polyclip_1.10-6             graph_1.74.0
##  [69] Matrix_1.6-5                ggseqlogo_0.2
##  [71] processx_3.8.4              GlobalOptions_0.1.2
##  [73] png_0.1-8                   rjson_0.2.21
##  [75] bitops_1.0-7                R.oo_1.26.0
##  [77] cmdfun_1.0.2                KernSmooth_2.23-22
##  [79] blob_1.2.4                  shape_1.4.6.1
##  [81] qvalue_2.28.0               regioneR_1.28.0
##  [83] scales_1.3.0                memoise_2.0.1
##  [85] magrittr_2.0.3              plyr_1.8.9
##  [87] gplots_3.1.3.1              zlibbioc_1.44.0
##  [89] compiler_4.2.1              BiocIO_1.6.0
##  [91] clue_0.3-65                 Rsamtools_2.14.0
##  [93] cli_3.6.2                   affy_1.76.0
##  [95] ps_1.7.6                    formatR_1.14
##  [97] MASS_7.3-60.0.1             mgcv_1.9-1
##  [99] tidyselect_1.2.1            vsn_3.66.0
## [101] stringi_1.8.3               highr_0.10
## [103] yaml_2.3.8                  locfit_1.5-9.9
## [105] ggstats_0.5.1               polynom_1.4-1
## [107] tools_4.2.1                 timechange_0.3.0
## [109] parallel_4.2.1              rstudioapi_0.15.0
## [111] foreach_1.5.2               farver_2.1.1
## [113] digest_0.6.35               BiocManager_1.30.22
## [115] shiny_1.8.0                 Rcpp_1.0.12
## [117] BiocVersion_3.15.2          later_1.3.2
## [119] httr_1.4.7                  colorspace_2.1-0
## [121] polylabelr_0.2.0            brio_1.1.4
## [123] XML_3.99-0.16.1             truncnorm_1.0-9
## [125] splines_4.2.1               RBGL_1.72.0
## [127] rematch2_2.1.2              confintr_1.0.2
## [129] multtest_2.52.0             bedtoolsr_2.30.0-4
## [131] xtable_1.8-4                futile.options_1.0.1
## [133] testthat_3.2.1              R6_2.5.1
## [135] pillar_1.9.0                htmltools_0.5.7
## [137] mime_0.12                   glue_1.7.0
## [139] fastmap_1.1.1               DT_0.32
## [141] BiocParallel_1.32.6         interactiveDisplayBase_1.34.0
## [143] codetools_0.2-19            utf8_1.2.4
## [145] lattice_0.22-5              curl_5.2.1
## [147] gtools_3.9.5                magick_2.8.2
## [149] survival_3.5-8              limma_3.54.2
## [151] rmarkdown_2.26              InteractionSet_1.24.0
## [153] desc_1.4.3                  munsell_0.5.0
## [155] GetoptLong_1.0.5            GenomeInfoDbData_1.2.9
## [157] iterators_1.0.14            reshape2_1.4.4
## [159] gtable_0.3.4
```