irCLIP-RNP dataset of HNRNPC during EGF stimulation

Luca Ducoli

28 June, 2024

This is the pipeline used to analyze the irCLIP-RNP TMT datasets of HNRNPC during EGF stimulation from two different gel sections ranging from 60-120kDa, 120-350kDa. The experiment was performed in A431 cells.

1. Prepare the dataset

```
#Needed libraries
library(DEP2)
library(tidyverse)
library(ggplot2)
library(data.table)
library(pheatmap)
library(RColorBrewer)
library(gplots)
library(hrbrthemes)
library(pacman)
library(textshape)
library(ggExtra)
library(viridis)
library(purrr)
library(hexbin)
library(DESeq2)
library(ggpubr)
library(UpSetR)
library(dplyr)
library(Clipper)
library(factoextra)
library(paletteer)
library(corrplot)
library(psych)
library(ggpmisc)
library(gprofiler2)
library(viridis)
library(GGally)
library(igraph)
library(rstatix)
library(limma)
library(HDMD)
library(cluster)
```

2. Determine the RDAPs

We used ClippeR to determined the significant RDAPs. We used a label-free HNRNPC irCLIP-RNP dataset coming from A431 (1 noUV and 2 UVC samples).

```
# Open proteinGroups.txt results from MaxQuant
data <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/0_Data/
    proteinGroups.txt")

#Remove RPL proteins
data <- data[-grep("RPL", data$Gene.names),]

#Generate unique names and ids
unique_pg <- make_unique(data, name = "Gene.names", ids = "Protein.IDs")
unique_pg <- unique_pg %>% arrange(name)

#Get the columns
ecols <- grep("LFQ.intensity.", colnames(unique_pg))

#Keep isoform with higher LFQ intensity
iso <- grep("\\.\\d+$", unique_pg$name)
rbp <- gsub("\\.1", "", c(unique_pg$name[iso]))

#Find original row name of the isoform with higher intensity
find_max_value <- function(rbp) {
  filtered_df <- unique_pg[unique_pg$name %like% rbp, grep("LFQ.intensity.", colnames(unique_pg))
      ]
  filtered_df$rowSums <- rowSums(filtered_df[, grep("LFQ.intensity.", colnames(filtered_df))])
  max_value <- which.max(filtered_df$rowSums)
  rownames <- rownames(filtered_df)[-max_value]
  return(rownames)
}
max_iso <- c(unlist(lapply(rbp, find_max_value)))

#Remove low intensity isoforms
unique_pg <- unique_pg[!(rownames(unique_pg) %in% max_iso),]

# Remove all proteins detected in IgG
unique_pg <- subset(unique_pg, LFQ.intensity.BZ101 == 0)

# Remove IgG column
unique_pg <- unique_pg[,-c(43)]
```

```
# Load design matrix
design <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/0_Data/
    Design_matrix.txt")
design
```

```
##                   label condition cell_type replicate    rbp
## 1 LFQ.intensity.BZ100       UVC      A431         1 HNRNPC
## 2  LFQ.intensity.BZ98      noUV      A431         1 HNRNPC
## 3  LFQ.intensity.BZ99       UVC      A431         2 HNRNPC
```

```
# Create a SummarizedExperiment
ecols <- grep("LFQ.intensity.", colnames(unique_pg))
se <- make_se(unique_pg, columns = ecols, expdesign = design)
se_UVC <- se[,se$condition == "UVC"]
se_UVC <- filter_se(se_UVC, thr = 0, filter_formula = ~ Reverse != '+' & Potential.contaminant !=
    "+" & Peptides > 1 & Unique.peptides > 0)
se <- se[rownames(se_UVC),]
write.table(se@assays@data@listData, file="~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_
    timecourse/2_A431_ClippeR/HNRNPC_A431_LFQ_intensity_raw.txt", quote = F, row.names = T, sep =
     "\t")

set.seed(3)
```

```
# Run Clipper
imputed <- DEP2::impute(normalize_vsn(se), fun = "QRILC")
data <- as.data.frame(assay(imputed))
clipper = Clipper(score.exp = as.matrix(data[,c(1,3)]), score.back = as.matrix(data[,-c(1,3)]),
    FDR = 0.05, analysis = "e")
data$FDR <- clipper$q
data <- cbind(data, rowMeans(data[,c(1,3)])-data[2])
colnames(data)[5] <- c("logFC")
write.table(data, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/2_A431_
    ClippeR/A431_Clipper_results.txt", quote = F, sep = "\t")
deg <- subset(data, FDR < 0.1 & logFC > log2(3))

deg
```

```
##                UVC_1    noUV_1    UVC_2        FDR     logFC
## AKAP8       20.84741 16.30699 20.93312 0.02272727  4.583276
## ANXA2       21.46484 16.20619 18.63407 0.02272727  3.843262
## CELF1       20.56323 18.17426 20.15149 0.06896552  2.183096
## CSDE1       20.63824 17.86557 20.68860 0.05357143  2.797852
## DDX17       22.18251 20.05268 22.28383 0.06896552  2.180494
## DDX3X       20.28579 17.57273 20.52661 0.05357143  2.833470
## DDX5        23.79769 21.16704 24.11827 0.05357143  2.790938
## DHX9        23.73612 18.69681 21.88923 0.02272727  4.115865
## ELAVL1      25.07109 14.25503 25.35966 0.02272727 10.960349
## ESRP1       20.38056 16.24174 20.87445 0.02272727  4.385761
## EWSR1       23.44545 16.31090 23.71751 0.02272727  7.270576
## FUBP1       21.39481 15.09730 21.96924 0.02272727  6.584725
## FUBP3       23.67288 15.11787 23.75030 0.02272727  8.593719
## FUS         25.11543 21.86715 24.77847 0.04166667  3.079801
## HNRNPA0     22.34423 18.36853 22.18053 0.02272727  3.893851
## HNRNPA1     25.00816 17.34484 25.02033 0.02272727  7.669408
## HNRNPA2B1 25.82773 15.63528 25.84789 0.02272727 10.202535
## HNRNPA3     23.84774 16.98427 23.62045 0.02272727  6.749830
## HNRNPAB     23.24761 17.99411 22.83719 0.02272727  5.048290
## HNRNPD      23.84988 18.37437 24.38017 0.02272727  5.740656
## HNRNPF      23.28187 18.36922 23.35438 0.02272727  4.948904
## HNRNPH2     19.63723 15.51536 19.40854 0.02272727  4.007524
## HNRNPH3     22.77165 17.55646 22.98032 0.02272727  5.319522
## HNRNPK      24.83347 16.83408 24.91489 0.02272727  8.040101
## HNRNPL      25.00235 18.50922 25.03208 0.02272727  6.507994
## HNRNPM      24.74329 16.78852 25.05747 0.02272727  8.111868
## HNRNPR      22.96935 18.57873 23.00191 0.02272727  4.406894
## HNRNPUL2    21.83514 16.25714 20.91162 0.02272727  5.116241
## IGF2BP2     20.43307 15.68218 20.25607 0.02272727  4.662383
## ILF2        19.73235 17.30558 19.84067 0.05357143  2.480926
## KHDRBS1     22.42169 14.66893 21.87733 0.02272727  7.480582
## KHSRP       23.79843 17.44869 23.82423 0.02272727  6.362643
## MATR3       23.82981 17.34540 23.83543 0.02272727  6.487222
## NCL         21.55702 15.97544 21.71208 0.02272727  5.659107
## NONO        22.36633 17.57314 22.45584 0.02272727  4.837948
## PCBP1       21.60640 18.19702 21.93319 0.02272727  3.572776
## PCBP2       21.30323 14.84525 21.23710 0.02272727  6.424912
## PKP1        22.63298 19.57835 21.87251 0.05357143  2.674398
## PRPF8       18.66399 15.94500 18.53990 0.05357143  2.656945
## PSPC1       20.00201 15.20082 20.54158 0.02272727  5.070977
## PTBP1       24.75513 14.61936 24.31776 0.02272727  9.917086
## RALY        25.18009 18.40393 25.39941 0.02272727  6.885819
## RBFOX2      18.57074 13.42687 18.67514 0.02272727  5.196064
## RBM14       21.92681 18.85646 22.09666 0.04166667  3.155278
## RBM15       20.47737 17.17749 20.48231 0.02272727  3.302352
## RBM4        20.83461 16.50711 20.95840 0.02272727  4.389400
## RBMX        21.78579 18.13136 21.75198 0.02272727  3.637521
## RPS2        22.57579 18.14959 23.14204 0.02272727  4.709330
## SAFB2       20.30931 17.34191 20.79650 0.04166667  3.210997
## SFPQ        24.06834 14.65922 24.05041 0.02272727  9.400153
## SRSF5       20.35028 17.27044 20.35968 0.04166667  3.084540
## SYNCRIP    24.59383 22.04686 24.86577 0.05357143  2.682935
## TARDBP     23.19232 18.23400 23.00559 0.02272727  4.864950
## U2SURP     20.28537 16.21283 19.34989 0.02272727  3.604805
## UPF1       19.05721 15.19660 18.38297 0.02272727  3.523498
## XRCC5      20.41072 14.52830 20.12166 0.02272727  5.737896
## YBX1       22.61712 15.91447 22.31557 0.02272727  6.551874
## ZNF326     20.58186 18.24665 21.64180 0.05357143  2.865182
```

3. Perform differential enrichement analysis

Here, we perform differential enrichment analysis of the TMT data using the DEP2 package.

```
# Open TMT data that were searched with MaxQuant and processed with Perseus
colnames <- c("name", "ID", "EGF0.R1.L",     "EGF0.R1.H",     "EGF0.R2.L",     "EGF0.R2.H",     "
     EGF15.R1.L",     "EGF15.R1.H",     "EGF15.R2.L",
                "EGF15.R2.H", "EGF30.R1.L",     "EGF30.R1.H",     "EGF30.R2.L",     "EGF30.R2.H",     "
     EGF60.R1.L",     "EGF60.R1.H",     "EGF60.R2.L",     "EGF60.R2.H")
EGF_data <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/0_TMT_data
     /EGF_BZ61.txt")
EGF_data$Gene.names <- str_match_all(EGF_data$Fasta.headers, "GN=(.*?) PE") %>% lapply(.,
     function (x) str_c(x[,2],collapse='; ')) %>% unlist()
EGF_data$Prot.IDs <- str_match_all(EGF_data$Fasta.headers, "(?<=sp\\|)[[:alnum:]]+") %>% lapply
     (., function (x) str_c(x[,1],collapse='; ')) %>% unlist()

#get the unique gene names and protein IDs
EGF_data$Gene.names %>% duplicated() %>% any() # check for duplicates
```

```
## [1] FALSE
```

```
EGF_data <- make_unique(EGF_data, "Gene.names", "Prot.IDs", delim = ";")
EGF_data$name %>% duplicated() %>% any() # must be false
```

```
## [1] FALSE
```

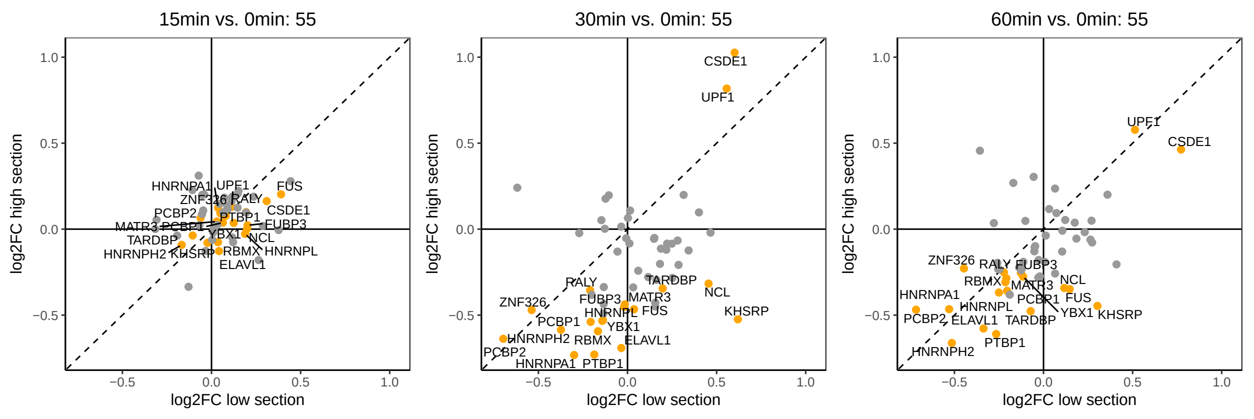
```
EGF_data <- EGF_data[,c(tail(grep("name", colnames(EGF_data) ), 1),tail(grep("ID", colnames(EGF_
     data) ), 1),grep("Reporter", colnames(EGF_data) ))]
colnames(EGF_data) <- colnames
EGF_data <- EGF_data[-grep("RPL", EGF_data$name),]
EGF_data$name[EGF_data$name == "RBFOX1"] <- "RBFOX2"
EGF_data$ID[EGF_data$ID == "Q9NWB1"] <- "O43251"

write.table(EGF_data, file ="~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/0_TMT
     _data/EGF_TMT_intensities.txt", row.names = FALSE, sep = '\t', quote = FALSE)

#design matrix
design <- read.delim("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/3_DEP/0_
     design.txt")

#generate columnns indexes
columns_vsn <- c(grep("EGF", colnames(EGF_data)))

# SummarizedExperiment
```

```r
EGF_data[,3:18] <- 2^EGF_data[,3:18]
EGF_se_UVC_norm_vsn <- make_se(EGF_data, columns_vsn, design)
EGF_se_UVC_norm_vsn <- normalize_vsn(EGF_se_UVC_norm_vsn)
write.table(as.data.frame(assay(EGF_se_UVC_norm_vsn)), file = "~/Documents/Postdoc/PD_Projects/3_
    irCLIP-RNP/MS/EGF_timecourse/3_DEP/EGF_TMT_LFQ_vsn.txt", row.names = TRUE, sep = "\t")

#get contrasts for each cell line and each normalization
model_vsn <- model.matrix(~ section + time:section, colData(EGF_se_UVC_norm_vsn))

#interaction analysis
EGF_fit1_norm_int_vsn = lmFit(assay(EGF_se_UVC_norm_vsn), design = model.matrix(~ section + time:
    section, colData(EGF_se_UVC_norm_vsn)))
EGF_fit2_norm_int_vsn <- eBayes(EGF_fit1_norm_int_vsn)
EGF_int_norm_vsn_both <- topTable(EGF_fit2_norm_int_vsn, coef = c("sectionhigh:timeT15","
    sectionhigh:timeT30", "sectionhigh:timeT60","sectionlow:timeT15","sectionlow:timeT30", "
    sectionlow:timeT60"), number = length(rownames(EGF_se_UVC_norm_vsn)))

EGF_sign_prot <- subset(EGF_int_norm_vsn_both, adj.P.Val < 0.1)
EGF_sign_prot <- subset(EGF_sign_prot,
                        abs(sectionhigh.timeT15) > 0.3 | abs(sectionhigh.timeT30) > 0.3 | abs(
    sectionhigh.timeT60) > 0.3
                        | abs(sectionlow.timeT15) > 0.3 | abs(sectionlow.timeT30) > 0.3 | abs(
    sectionlow.timeT60) > 0.3 )

EGF_int_norm_vsn_both$gene <- rownames(EGF_int_norm_vsn_both)
write.table(EGF_int_norm_vsn_both, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_
    timecourse/3_DEP/EGF_TMT_res_norm_vsn_DEP_int.txt", row.names = FALSE, sep = "\t")
```

Scatterplot of significant RDAPs

```r
#Get significance labels for scatterplot
EGF_int_norm_vsn_both <- subset(EGF_int_norm_vsn_both, rownames(EGF_int_norm_vsn_both) %in%
    rownames(deg))
EGF_int_norm_vsn_both$int_sign <- EGF_int_norm_vsn_both$gene %in% rownames(EGF_sign_prot)
EGF_colors <- c("FALSE" = "#999999","TRUE" =  "orange")

EGF_vsn_max_axis <- c(max(c( max(EGF_int_norm_vsn_both$sectionlow.timeT15, EGF_int_norm_vsn_both$
    sectionhigh.timeT15),max(EGF_int_norm_vsn_both$sectionlow.timeT30, EGF_int_norm_vsn_both$
    sectionhigh.timeT30), max(EGF_int_norm_vsn_both$sectionlow.timeT60, EGF_int_norm_vsn_both$
    sectionhigh.timeT60))))
EGF_vsn_min_axis <- c(max(abs(min(EGF_int_norm_vsn_both$sectionlow.timeT15, EGF_int_norm_vsn_both
    $sectionhigh.timeT15)),abs(min(EGF_int_norm_vsn_both$sectionlow.timeT30, EGF_int_norm_vsn_
    both$sectionhigh.timeT30)), abs(min(EGF_int_norm_vsn_both$sectionlow.timeT60, EGF_int_norm_
    vsn_both$sectionhigh.timeT60))))

ggplot.15min <- ggplot(data=EGF_int_norm_vsn_both, aes(x=sectionlow.timeT15, y=sectionhigh.
    timeT15)) + geom_vline(xintercept = 0) + geom_hline(yintercept = 0) +  geom_abline(intercept
    = 0, linetype=2) +
  geom_point(shape=19, size=2, aes(col = int_sign)) +
  labs(title = paste("15min vs. 0min:", nrow(EGF_int_norm_vsn_both))   , x = expression("log2FC
    low section"), y = expression("log2FC high section")) +
  scale_color_manual(values = EGF_colors) +
  ggrepel::geom_text_repel(data = EGF_int_norm_vsn_both[EGF_int_norm_vsn_both$int_sign == "TRUE"
    ,], aes(label = gene), size = 3, box.padding = unit(0.1, "lines"), point.padding = unit(0.1,
    "lines"), segment.size = 0.5,max.overlaps = Inf) +
  theme_bw() +
  theme(legend.position = "none", panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5)) +
  xlim(-EGF_vsn_min_axis, EGF_vsn_max_axis) +
  ylim(-EGF_vsn_min_axis, EGF_vsn_max_axis)
```

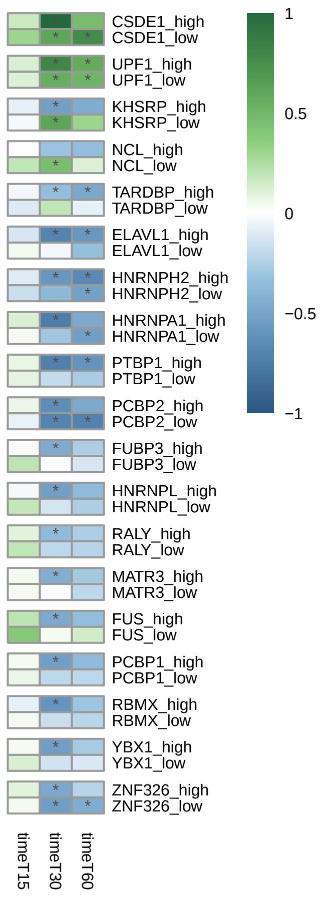
```
ggplot.30min <- ggplot(data=EGF_int_norm_vsn_both, aes(x=sectionlow.timeT30, y=sectionhigh.
    timeT30)) + geom_vline(xintercept = 0) + geom_hline(yintercept = 0) +  geom_abline(intercept
    = 0, linetype=2) +
  geom_point(shape=19, size=2, aes(col = int_sign)) +
  labs(title = paste("30min vs. 0min:", nrow(EGF_int_norm_vsn_both))   , x = expression("log2FC
    low section"), y = expression("log2FC high section")) +
  scale_color_manual(values = EGF_colors) +
  ggrepel::geom_text_repel(data = EGF_int_norm_vsn_both[EGF_int_norm_vsn_both$int_sign == "TRUE"
    ,], aes(label = gene), size = 3, box.padding = unit(0.1, "lines"), point.padding = unit(0.1,
    "lines"), segment.size = 0.5,max.overlaps = Inf) +
  theme_bw() +
  theme(legend.position = "none", panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5)) +
  xlim(-EGF_vsn_min_axis, EGF_vsn_max_axis) +
  ylim(-EGF_vsn_min_axis, EGF_vsn_max_axis)

ggplot.60min <- ggplot(data=EGF_int_norm_vsn_both, aes(x=sectionlow.timeT60, y=sectionhigh.
    timeT60)) + geom_vline(xintercept = 0) + geom_hline(yintercept = 0) +  geom_abline(intercept
    = 0, linetype=2) +
  geom_point(shape=19, size=2, aes(col = int_sign)) +
  labs(title = paste("60min vs. 0min:", nrow(EGF_int_norm_vsn_both))   , x = expression("log2FC
    low section"), y = expression("log2FC high section")) +
  scale_color_manual(values = EGF_colors) +
  ggrepel::geom_text_repel(data = EGF_int_norm_vsn_both[EGF_int_norm_vsn_both$int_sign == "TRUE"
    ,], aes(label = gene), size = 3, box.padding = unit(0.1, "lines"), point.padding = unit(0.1,
    "lines"), segment.size = 0.5,max.overlaps = Inf) +
  theme_bw() +
  theme(legend.position = "none", panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5)) +
  xlim(-EGF_vsn_min_axis, EGF_vsn_max_axis) +
  ylim(-EGF_vsn_min_axis, EGF_vsn_max_axis)


ggarrange(ggplot.15min ,ggplot.30min, ggplot.60min, ncol = 3)
```



```
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/4_Visualization/EGF_DEP_
    scatter_vsn_plot.pdf", height = 6, width = 18)
ggarrange(ggplot.15min ,ggplot.30min, ggplot.60min, ncol = 3)
dev.off()
```

Heatmap of significant RDAPs

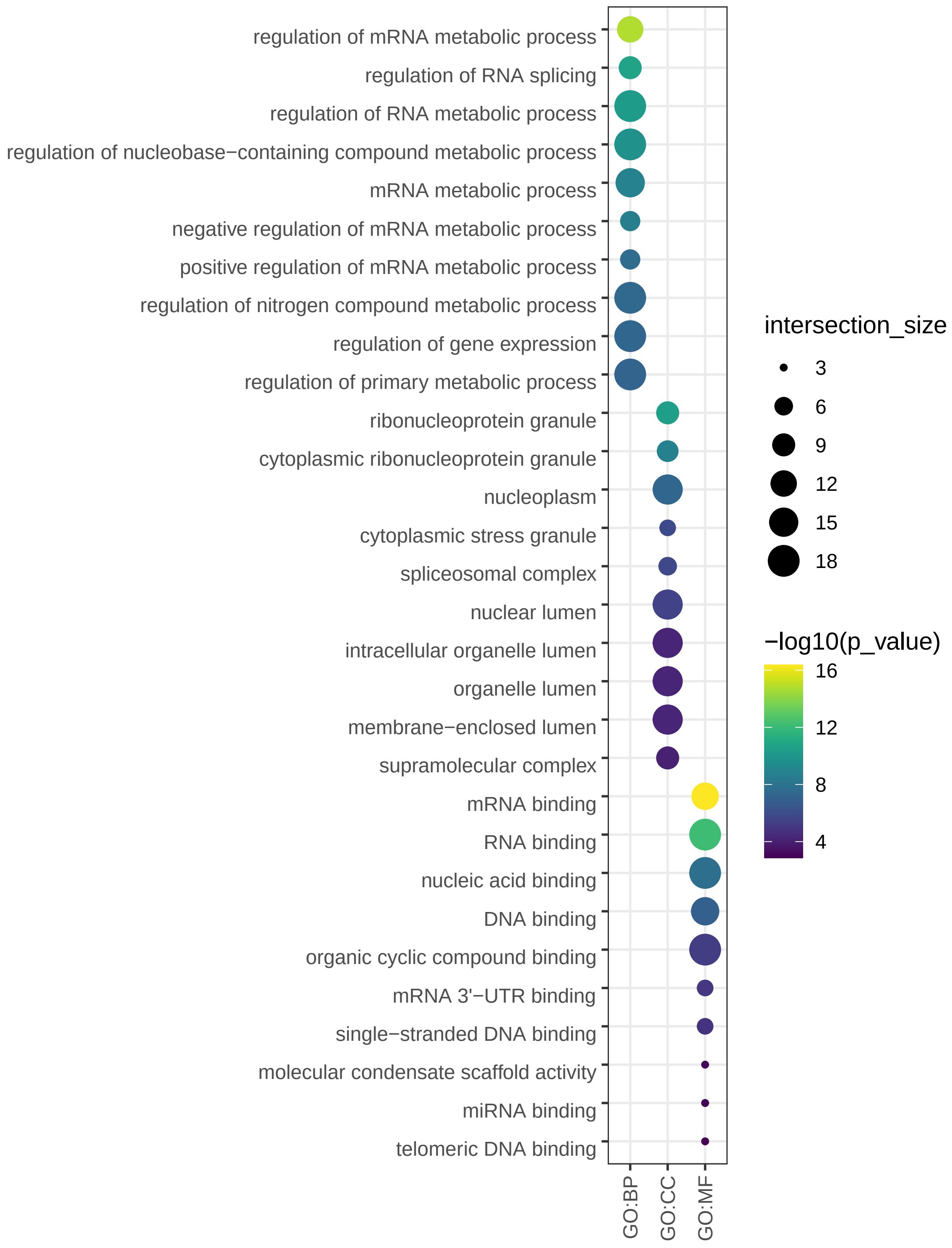
```
#Heatmap
EGF_int_norm_vsn_15low <- topTable(EGF_fit2_norm_int_vsn, coef = c("sectionlow:timeT15"), number
    = length(rownames(EGF_se_UVC_norm_vsn)))
EGF_sign_15low <- subset(EGF_int_norm_vsn_15low,  adj.P.Val < 0.1 & abs(logFC) > 0.3)
EGF_int_norm_vsn_15high <- topTable(EGF_fit2_norm_int_vsn, coef = c("sectionhigh:timeT15"),
    number = length(rownames(EGF_se_UVC_norm_vsn)))
EGF_sign_15high <- subset(EGF_int_norm_vsn_15high,  adj.P.Val < 0.1 & abs(logFC) > 0.3)

EGF_int_norm_vsn_30low <- topTable(EGF_fit2_norm_int_vsn, coef = c("sectionlow:timeT30"), number
    = length(rownames(EGF_se_UVC_norm_vsn)))
EGF_sign_30low <- subset(EGF_int_norm_vsn_30low,  adj.P.Val < 0.1 & abs(logFC) > 0.3)
EGF_int_norm_vsn_30high <- topTable(EGF_fit2_norm_int_vsn, coef = c("sectionhigh:timeT30"),
    number = length(rownames(EGF_se_UVC_norm_vsn)))
EGF_sign_30high <- subset(EGF_int_norm_vsn_30high,  adj.P.Val < 0.1 & abs(logFC) > 0.3)

EGF_int_norm_vsn_60low <- topTable(EGF_fit2_norm_int_vsn, coef = c("sectionlow:timeT60"), number
    = length(rownames(EGF_se_UVC_norm_vsn)))
EGF_sign_60low <- subset(EGF_int_norm_vsn_60low,  adj.P.Val < 0.1 & abs(logFC) > 0.3)
EGF_int_norm_vsn_60high <- topTable(EGF_fit2_norm_int_vsn, coef = c("sectionhigh:timeT60"),
    number = length(rownames(EGF_se_UVC_norm_vsn)))
EGF_sign_60high <- subset(EGF_int_norm_vsn_60high,  adj.P.Val < 0.1 & abs(logFC) > 0.3)

lt.tsk = list(T15_low = rownames(EGF_sign_15low),
              T15_high = rownames(EGF_sign_15high),
              T30_low = rownames(EGF_sign_30low),
              T30_high = rownames(EGF_sign_30high),
              T60_low = rownames(EGF_sign_60low),
              T60_high = rownames(EGF_sign_60high))

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)
  row.names(data) <- elements
  return(data)
}

#Binary table with colnames:
sign.proteins <- fromList(lt.tsk)
sign.proteins <- subset(sign.proteins, rownames(sign.proteins) %in% rownames(EGF_int_norm_vsn_
    both)[EGF_int_norm_vsn_both$int_sign == TRUE])
write.table(sign.proteins, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse
    /4_Visualization/EGF_sign_int_upset.txt", row.names = TRUE, sep = "\t")

#Matrix
EGF_sign_prot_HM <- EGF_sign_prot[,c(1,4,2,5,3,6)]
EGF_sign_prot_HM <- subset(EGF_sign_prot_HM, rownames(EGF_sign_prot_HM) %in% rownames(EGF_int_
    norm_vsn_both)[EGF_int_norm_vsn_both$int_sign == TRUE])

EGF_sign_prot_HM$name <- rownames(EGF_sign_prot_HM)
EGF_sign_prot_HM <- EGF_sign_prot_HM %>%
  pivot_longer(
    cols = colnames(EGF_sign_prot_HM)[1:6],
    names_to = "time_section",
    values_to = "logFC"
  )

EGF_sign_prot_HM <- EGF_sign_prot_HM %>% separate(time_section, c('section', 'time'))

EGF_sign_prot_HM <- EGF_sign_prot_HM %>%
  pivot_wider(
    names_from = c(time),
    values_from = logFC
  )

group = matrix(EGF_sign_prot_HM$name)
group = t(group[,1])

#Mahala clustering
variables = c("timeT15", "timeT30", "timeT60")
variables = as.matrix(EGF_sign_prot_HM[,variables])
mahala_sq = pairwise.mahalanobis(x=variables, grouping=group)
names = rownames(mahala_sq$means)

mahala = sqrt(mahala_sq$distance)
rownames(mahala) = names
colnames(mahala) = names

cluster = agnes(mahala,diss=TRUE,keep.diss=FALSE,method="ward")

orders <- rownames(mahala)[cluster$order]
orders <- rep(orders, each = 2)
match <- match(orders, EGF_sign_prot_HM$name)

match[c(1:length(match))[lapply(c(1:length(match)), "%%", 2) == 0]] <- match[c(1:length(match))[
     lapply(c(1:length(match)), "%%", 2) == 0]]+1

EGF_sign_prot_HM2 <- EGF_sign_prot_HM[match,]

EGF_sign_prot_HM_mx <- as.matrix(EGF_sign_prot_HM2[3:5])
rownames(EGF_sign_prot_HM_mx) <- paste(EGF_sign_prot_HM2$name, EGF_sign_prot_HM2$section, sep = "
     _")
rownames(EGF_sign_prot_HM_mx) <- sub("section", "", rownames(EGF_sign_prot_HM_mx))

#Color palette and annotation
my.breaks <- c(seq(-1, 1, by=0.01))
my.colors <- c(rev(paletteer_c("ggthemes::Green-Blue-White Diverging", length(my.breaks))))

sign.proteins$name <- rownames(sign.proteins)
sign.proteins2 <- sign.proteins %>%
  pivot_longer(
    cols = colnames(sign.proteins)[1:6],
    names_to = "time_section",
    values_to = "sign"
  )

sign.proteins2 <- sign.proteins2 %>% separate(time_section, c('time', 'section'))

sign.proteins2 <- sign.proteins2 %>%
  pivot_wider(
    names_from = c(time),
    values_from = sign
  )

sign.proteins2 <- as.data.frame(sign.proteins2)
rownames(sign.proteins2) <- paste(sign.proteins2$name, sign.proteins2$section, sep = "_")
sign.proteins3 <- sign.proteins2[rownames(sign.proteins2) %in% rownames(EGF_sign_prot_HM_mx),]
labels <- sign.proteins3[,c(3:5)]
colnames(labels) <- colnames(EGF_sign_prot_HM_mx)
labels[labels == 1] <- "*"
labels[labels == 0] <- ""
labels <- labels[match(rownames(EGF_sign_prot_HM_mx), rownames(labels)),]

#Heatmap
pheatmap <- pheatmap(
  mat              = EGF_sign_prot_HM_mx,
  cellwidth = 12,
  cellheight = 6,
  display_numbers = labels,
  fontsize_number=5.5,
  color = my.colors,
  breaks = my.breaks,
  show_colnames      = TRUE,
  show_rownames      = TRUE,
  drop_levels        = TRUE,
  fontsize           = 5.5,
  cluster_rows       = FALSE,
  cluster_cols       = FALSE,
  gaps_row = c(1:length(rownames(EGF_sign_prot_HM_mx)))[lapply(c(1:length(rownames(EGF_sign_prot_
    HM_mx))), "%%", 2) == 0],
)
```



```
#Save heatmap
pheatmap <- pheatmap(
  mat                    = EGF_sign_prot_HM_mx,
   cellwidth = 12,
   cellheight = 6,
   display_numbers = labels ,
   fontsize_number=5.5,
   color = my.colors ,
   breaks = my.breaks ,
  show_colnames          = TRUE,
  show_rownames          = TRUE,
   drop_levels           = TRUE,
   fontsize              = 5.5 ,
   cluster_rows          = FALSE,
  cluster_cols        = FALSE,
  gaps_row = c(1:length(rownames(EGF_sign_prot_HM_mx)))[lapply(c(1:length(rownames(EGF_sign_prot_
    HM_mx))), "%%", 2) == 0],
  filename = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/4_Visualization/EGF_
    sign_HM.pdf",
  width = 5,
  height = 7
)

pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/4_Visualization/EGF_sign_
    dendo.pdf", height = 5, width = 10)
plot(cluster,which.plots=2, hang = -1)
dev.off()
```

Gene ontology analysis

```
set.seed(3)
#Run GO analysis on UVC data
gp.res = gost(rownames(EGF_int_norm_vsn_both)[EGF_int_norm_vsn_both$int_sign == TRUE], organism =
     "hsapiens")

#Take top 20 terms for each source
gp.res <- gp.res$result %>% group_by(source) %>% dplyr::slice(1:10)
gp.bp <- gp.res[gp.res$source %in% c("GO:BP", "GO:CC", "GO:MF"),]
gp.bp$term_name <- factor(gp.bp$term_name, levels = unique(gp.bp$term_name))
gp.bp$source <- factor(gp.bp$source, levels = unique(gp.bp$source))

#Prepare the bubble plot
ggplot(data = gp.bp, aes(x = source, y = term_name, color =-log10(p_value), size = intersection_
    size)) +
  geom_point() +
  scale_color_viridis(option = "D") +
  theme_bw() +
  ylab("") +
  xlab("") +
  theme(axis.text.y = element_text(vjust = 1, hjust=1), axis.text.x = element_text(angle = 90,
    vjust = 0.5, hjust=1))+
  scale_y_discrete(limits=rev)
```



```
# Save the bubble plot as pdf
pdf("~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/4_Visualization/GO_EGF_
    protein.pdf", height = 8, width = 6.25)
ggplot(data = gp.bp, aes(x = source, y = term_name, color = -log10(p_value), size = intersection_
    size)) +
  geom_point() +
  scale_color_viridis(option = "D") +
  theme_bw() +
  ylab("") +
  xlab("") +
  theme(axis.text.y = element_text(vjust = 1, hjust=1), axis.text.x = element_text(angle = 90,
    vjust = 0.5, hjust=1))+
  scale_y_discrete(limits=rev)
dev.off()
```

```
#Get all results per time point
colnames(EGF_int_norm_vsn_15low) <- str_c("15min_low_", colnames(EGF_int_norm_vsn_15low))
colnames(EGF_int_norm_vsn_15high) <- str_c("15min_high_", colnames(EGF_int_norm_vsn_15high))
colnames(EGF_int_norm_vsn_30low) <- str_c("30min_low_", colnames(EGF_int_norm_vsn_30low))
colnames(EGF_int_norm_vsn_30high) <- str_c("30min_high_", colnames(EGF_int_norm_vsn_30high))
colnames(EGF_int_norm_vsn_60low) <- str_c("60min_low_", colnames(EGF_int_norm_vsn_60low))
colnames(EGF_int_norm_vsn_60high) <- str_c("60min_high_", colnames(EGF_int_norm_vsn_60high))

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

all <- merge.all(EGF_int_norm_vsn_15low,EGF_int_norm_vsn_15high,EGF_int_norm_vsn_30low,EGF_int_
    norm_vsn_30high,EGF_int_norm_vsn_60low,EGF_int_norm_vsn_60high)
write.table(all, file = "~/Documents/Postdoc/PD_Projects/3_irCLIP-RNP/MS/EGF_timecourse/3_DEP/EGF
    _TMT_contrast.txt", row.names = TRUE, sep = "\t")
```

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] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] cluster_2.1.6              HDMD_1.2
##  [3] MASS_7.3-60.0.1            rstatix_0.7.2
##  [5] igraph_2.0.3               GGally_2.2.1
##  [7] gprofiler2_0.2.3           ggpmisc_0.5.5
##  [9] ggpp_0.5.6                 psych_2.4.3
## [11] corrplot_0.92              paletteer_1.6.0
## [13] factoextra_1.0.7           Clipper_0.0.0.9000
## [15] UpSetR_1.4.0               ggpubr_0.6.0
## [17] DESeq2_1.38.3              hexbin_1.28.3
## [19] viridis_0.6.5              viridisLite_0.4.2
## [21] ggExtra_0.10.1                textshape_1.7.3
## [23] pacman_0.5.1                  hrbrthemes_0.8.7
## [25] gplots_3.1.3.1                RColorBrewer_1.1-3
## [27] pheatmap_1.0.12               data.table_1.15.2
## [29] lubridate_1.9.3               forcats_1.0.0
## [31] stringr_1.5.1                 dplyr_1.1.4
## [33] purrr_1.0.2                   readr_2.1.5
## [35] tidyr_1.3.1                   tibble_3.2.1
## [37] ggplot2_3.5.0                 tidyverse_2.0.0
## [39] DEP2_0.4.8.24                 R6_2.5.1
## [41] limma_3.54.2                  MSnbase_2.24.2
## [43] ProtGenerics_1.30.0           mzR_2.32.0
## [45] Rcpp_1.0.12                   MsCoreUtils_1.10.0
## [47] SummarizedExperiment_1.28.0 Biobase_2.58.0
## [49] GenomicRanges_1.50.2          GenomeInfoDb_1.34.9
## [51] IRanges_2.32.0                S4Vectors_0.36.2
## [53] BiocGenerics_0.44.0           MatrixGenerics_1.10.0
## [55] matrixStats_1.2.0
##
## loaded via a namespace (and not attached):
##   [1] SparseM_1.81                  ggthemes_5.1.0
##   [3] missForest_1.5                bit64_4.0.5
##   [5] knitr_1.45                    DelayedArray_0.24.0
##   [7] KEGGREST_1.38.0               RCurl_1.98-1.14
##   [9] AnnotationFilter_1.22.0       doParallel_1.0.17
##  [11] generics_0.1.3                preprocessCore_1.60.2
##  [13] cowplot_1.1.3                 RSQLite_2.3.5
##  [15] proxy_0.4-27                  bit_4.0.5
##  [17] tzdb_0.4.0                    httpuv_1.6.14
##  [19] assertthat_0.2.1              TCseq_1.22.6
##  [21] xfun_0.42                     hms_1.1.3
##  [23] evaluate_0.23                 promises_1.2.1
##  [25] fansi_1.0.6                   caTools_1.18.2
##  [27] htmlwidgets_1.6.4             DBI_1.2.2
##  [29] geneplotter_1.76.0            ellipsis_0.3.2
##  [31] RSpectra_0.16-1               QFeatures_1.8.0
##  [33] backports_1.4.1               fontLiberation_0.1.0
##  [35] prismatic_1.1.1               annotate_1.76.0
##  [37] fontBitstreamVera_0.1.1       vctrs_0.6.5
##  [39] imputeLCMD_2.1                quantreg_5.97
##  [41] abind_1.4-5                   cachem_1.0.8
##  [43] withr_3.0.0                   itertools_0.1-3
##  [45] GenomicAlignments_1.34.1      fdrtool_1.2.17
##  [47] MultiAssayExperiment_1.24.0 mnormt_2.1.1
##  [49] lazyeval_0.2.2                crayon_1.5.2
##  [51] crul_1.4.0                    labeling_0.4.3
##  [53] glmnet_4.1-8                  edgeR_3.40.2
##  [55] pkgconfig_2.0.3               nlme_3.1-164
##  [57] rlang_1.1.3                   lifecycle_1.0.4
##  [59] miniUI_0.1.1.1                sandwich_3.1-0
##  [61] MatrixModels_0.5-3            downloader_0.4
##  [63] fontquiver_0.2.1              httpcode_0.3.0
##  [65] affyio_1.68.0                 extrafontdb_1.0
##  [67] randomForest_4.7-1.1          rngtools_1.5.2
##  [69] Matrix_1.6-5                  carData_3.0-5
##  [71] zoo_1.8-12                    GlobalOptions_0.1.2
##  [73] png_0.1-8                     rjson_0.2.21
##  [75] bitops_1.0-7                  KernSmooth_2.23-22
##  [77] Biostrings_2.66.0             blob_1.2.4
##  [79] doRNG_1.8.6                   shape_1.4.6.1
##  [81] tmvtnorm_1.6                  ggsignif_0.6.4
##  [83] scales_1.3.0                  memoise_2.0.1
##  [85] magrittr_2.0.3                plyr_1.8.9
##  [87] zlibbioc_1.44.0               compiler_4.2.1
##  [89] pcaMethods_1.90.0             clue_0.3-65
##  [91] Rsamtools_2.14.0              cli_3.6.2
##  [93] affy_1.76.0                   XVector_0.38.0
##  [95] tidyselect_1.2.1              vsn_3.66.0
##  [97] stringi_1.8.3              highr_0.10
##  [99] yaml_2.3.8                 norm_1.0-11.1
## [101] askpass_1.2.0              locfit_1.5-9.9
## [103] MALDIquant_1.22.2          ggrepel_0.9.5
## [105] grid_4.2.1                 ggstats_0.5.1
## [107] polynom_1.4-1              tools_4.2.1
## [109] timechange_0.3.0           parallel_4.2.1
## [111] circlize_0.4.16            rstudioapi_0.15.0
## [113] foreach_1.5.2              gridExtra_2.3
## [115] farver_2.1.1               mzID_1.36.0
## [117] Rtsne_0.17                 digest_0.6.35
## [119] BiocManager_1.30.22        shiny_1.8.0
## [121] gfonts_0.2.0               car_3.1-2
## [123] broom_1.0.5                later_1.3.2
## [125] ncdf4_1.22                 httr_1.4.7
## [127] gdtools_0.3.5              AnnotationDbi_1.60.2
## [129] ComplexHeatmap_2.14.0      colorspace_2.1-0
## [131] XML_3.99-0.16.1            reticulate_1.35.0
## [133] umap_0.2.10.0              splines_4.2.1
## [135] rematch2_2.1.2             gmm_1.8
## [137] plotly_4.10.4              systemfonts_1.0.5
## [139] xtable_1.8-4               jsonlite_1.8.8
## [141] pillar_1.9.0               htmltools_0.5.7
## [143] mime_0.12                  glue_1.7.0
## [145] fastmap_1.1.1              BiocParallel_1.32.6
## [147] class_7.3-22               codetools_0.2-19
## [149] mvtnorm_1.2-4              utf8_1.2.4
## [151] lattice_0.22-5             curl_5.2.1
## [153] gtools_3.9.5               openssl_2.1.1
## [155] Rttf2pt1_1.3.12            survival_3.5-8
## [157] rmarkdown_2.26             munsell_0.5.0
## [159] e1071_1.7-14               GetoptLong_1.0.5
## [161] GenomeInfoDbData_1.2.9     iterators_1.0.14
## [163] impute_1.72.3              reshape2_1.4.4
## [165] gtable_0.3.4               extrafont_0.19
```