Differential binding of AGO in mir181KO vs WT

Mirko Brüggemann, Melina Klostermann

13 September, 2023

Contents

Libraries and settings

```
# ----------------------------------------
# libraries
# ----------------------------------------

library(rtracklayer)
library(GenomicRanges)
library(ggplot2)
library(AnnotationDbi)
library(dplyr)
library(reshape2)
library(UpSetR)
library(GenomicFeatures)
library(kableExtra)
library(knitr)
library(ggrepel)
library(gridExtra)
library(grid)
library(viridis)
library(BindingSiteFinder)
library(ComplexHeatmap)
library(forcats)
library(ggtext)
library(patchwork)
library(tibble)
```

```
library(tidyr)
library(dplyr)
library(ggpointdensity)
library(ggsci)
library(ggtext)
library(ggrepel)
library(patchwork)
library(ggrastr)
library(matrixStats)
library(DESeq2)
library(IHW)
library(ggrepel) # nikita

here <- here::here()

source(paste0(here,"/Supporting_scripts/themes/theme_paper.R"))
source(paste0(here,"/Supporting_scripts/themes/CustomThemes.R"))
source(paste0(here,"/Supporting_scripts/themes/colorPalette.R"))

#nikita
# source("D:/Krueger_Lab/Publications/miR181_paper/Supporting_scripts/themes/CustomThemes.R")
# source("D:/Krueger_Lab/Publications/miR181_paper/Supporting_scripts/themes/theme_paper.R")
# source("D:/Krueger_Lab/Publications/miR181_paper/Supporting_scripts/themes/colorPalette.R")

# ----------------------------------------
# settings
# ----------------------------------------

out <- paste0(here,"/Figure2/04_Differential_Binding/")
#nikita out
# out <- "D:/Krueger_Lab/Publications/miR181_paper/Figure1/Differential_Binding/"


tpm_cut <- 50
```

```
# ----------------------------------------
# ----------------------------------------
# Files
# ----------------------------------------
# ----------------------------------------

# ----------------------------------------
# annotation
# ----------------------------------------

annoDb <- readRDS(paste0(here, "/Supporting_scripts/annotation_preprocessing/annotation.rds"))
annoDb <- makeTxDbFromGRanges(annoDb)

gns <- readRDS(paste0(here, "/Supporting_scripts/annotation_preprocessing/gene_annotation.rds"))
#Nikita
# annoDb <- loadDb("D:/Krueger_Lab/Publications/miR181_paper_nongithub/Figure1/annotation.db")
# gns <- readRDS("D:/Krueger_Lab/Publications/miR181_paper/Methods/01_Annotation_preprocessing/gene_ann

# nikita source
```

```
# ----------------------------------------
# AGO binding sites
# ----------------------------------------

bs_wt <- readRDS(paste0(here, "/Figure1/01_AGO_binding_site_definition/AGO_BS.rds"))

bs_ko <- readRDS(paste0(here, "/Figure2/03_KOmir181_AGO_binding_site_definition/KOmir181_AGO_BS.rds"))

#nikita source
# bs_wt <- readRDS("D:/Krueger_Lab/Publications/miR181_paper/Figure1/AGO_binding_site_definition/2023-0
#
# bs_ko <- readRDS("D:/Krueger_Lab/Publications/miR181_paper/Methods/03_KOmir181_AGO_binding_site_defin

# ----------------------------------------
# Load clip data
# ----------------------------------------

clipFiles = "/Users/melinaklostermann/Documents/projects/AgoCLIP_miR181/pipe_output_22_02_14/non-chimer

#nikita source
# clipFiles = "D:/Krueger_Lab/miReCLIP/Melina/pipe_output_22_02_14/pipe_output_22_02_14/non-chimeric/bw

clipFiles = list.files(clipFiles, pattern = ".bw$", full.names = TRUE)
clipFiles = clipFiles[!grepl("Inp", clipFiles)]
clipFiles = clipFiles[!grepl("miR181_", clipFiles)]
clipFilesP = clipFiles[grep(clipFiles, pattern = "plus")]
clipFilesM = clipFiles[grep(clipFiles, pattern = "minus")]
```

farben

```
farbeneg <- "#B4B4B4"
farbe1 <- "#0073C2FF"
farbe3 <- "#CD534CFF" #miR181KO farbe
```

What was done?

Combine binding sites

Binding sites from both conditions do either overlap exactly, overlap partially or don't overlap at all. In the following partial overlaps are resolved by re-centering the binding sites based on the highest crosslink signal of both conditions.

```
# merge BS from both conditions
bsMerge = unique(c(resize(bs_wt, fix = "center", width = 1),
                   resize(bs_ko, fix = "center", width = 1)))

# Organize clip data in dataframe
colData = data.frame(
  id = c(1:6),
  condition = factor(c(rep("KO",3), rep("WT",3)),
                     levels = c("KO", "WT")),
```

Table 1: Merge and combine

Option	nRanges
inputRanges	37,575
mergeCrosslinkSites	28,346
minCrosslinks	28,346
minClSites	28,346
centerIsClSite	28,253
centerIsSummit	NA

```
                      clPlus = clipFilesP,
                      clMinus = clipFilesM)

# Make BindingSiteFinder object
bds = BSFDataSetFromBigWig(ranges = bsMerge, meta = colData)


# Make unified binding sites from both conditions
bdsMerge <- makeBindingSites(object = bds, bsSize = 7, minWidth = 0,
                         minCrosslinks = 0, minClSites = 0, centerIsSummit = FALSE)

df = getSummary(bdsMerge)
kable(myNumberFormat(df), caption = "Merge and combine")
```

Combination of both sets of binding sites results in a single set (N=28,253), where each binding site was either seen in *WT*, *KO* or both conditions. The figure below shows the set sizes in more detail.

Annotate combined binding sites with *PureCLIP score*, *WT* and *KO* status information.

```
# annotate with pureclip score
rng = getRanges(bds)
rng$additionalScores = NULL
mcols(rng)$score = rng$scoreSum
bdsFinal = annotateWithScore(bdsMerge, rng)

# annotate with condition support
rngFinal = getRanges(bdsFinal)
r1 = resize(bs_wt, fix = "center", width = 1)
r2 = resize(bs_ko, fix = "center", width = 1)
condition_support = data.frame(WT = countOverlaps(rngFinal, r1),
                KO = countOverlaps(rngFinal, r2))
mcols(rngFinal) = cbind(mcols(rngFinal), condition_support)

# transfer the geneID
olsWT = findOverlaps(rngFinal, bs_wt)
olsKO = findOverlaps(rngFinal, bs_ko)
rngFinal$geneID = NA
rngFinal$geneID[queryHits(olsWT)] = bs_wt$geneID[subjectHits(olsWT)]
rngFinal$geneID[queryHits(olsKO)] = bs_ko$geneID[subjectHits(olsKO)]

# transfer gene names
rngFinal$geneName = NA
rngFinal$geneName[queryHits(olsWT)] = bs_wt$geneName[subjectHits(olsWT)]
```

```
rngFinal$geneName[queryHits(olsKO)] = bs_ko$geneName[subjectHits(olsKO)]

# transfer transcript region
rngFinal$region = NA
rngFinal$region[queryHits(olsWT)] = bs_wt$region[subjectHits(olsWT)]
rngFinal$region[queryHits(olsKO)] = bs_ko$region[subjectHits(olsKO)]

# set final range
bdsFinal = setRanges(bdsFinal, rngFinal)
```

Perform Differntial binding

In order to perform differential binding analysis, we use DEseq2. The design formula contains both the number of crosslinks per binding site and the background crosslinks in the whole gene of the binding site.

Calculate crosslinks in bs and in gene background

The number of background crosslinks per gene is calculated to infer upregulation or downregulation of genes between the two conditions. The background contains all crosslinks on a gene except crosslinks in a binding site or within a 5nt offset to the binding sites.

```
### ============================================================================
### Get crosslink numbers in background and binding sites
### ----------------------------------------------------------------------------
# Function get coverage per bs
#------------------------
# NOTE: this is a simple copy from the BindingSiteFinder package, that uses
# the range and signal objects directly as input.
coverageBySignal <- function(range, signal,
         merge = TRUE,
         returnType = c("GRanges", "matrix", "data.frame")) {

    # split by strand
    rng = range
    rngPlus = rng[strand(rng) == "+"]
    rngMinus = rng[strand(rng) == "-"]
    # prepare signal
    sgn = signal

    # signal coverage is reported for each position in the range of the peak
    if (!isTRUE(merge)) {
        # manage return type
        # only return type data.frame is possible with this option
        returnType = match.arg(returnType,
                                choices = c("GRanges", "matrix", "data.frame"))
        if (returnType != "data.frame") {
            warning("Only return type 'data.frame' possible with non-merged output.")
        }
        returnType = "data.frame"

        if (length(rngPlus) > 0) {
            matPlus = lapply(sgn$signalPlus, function(x) {
                as.matrix(x[rngPlus])
```

```
            })
            covPlus = do.call(rbind, lapply(matPlus, colSums))
        }
        if (length(rngPlus) == 0) {
            covPlus = 0
        }
        if (length(rngMinus) > 0) {
            matMinus = lapply(sgn$signalMinus, function(x) {
                as.matrix(x[rngMinus])
            })
            covMinus = do.call(rbind, lapply(matMinus, colSums))
            # flip orientation of minus strand coverage
            covMinus = covMinus %>% as.data.frame() %>% rev() %>% as.matrix()
        }
        if (length(rngMinus) == 0) {
            covMinus = 0
        }
        covDf = covPlus + covMinus
        retObj = as.data.frame(covDf)
    }
    # signal is merged over all positions in the range
    if (isTRUE(merge)) {
        mcols(rngPlus) = as.matrix(
            do.call(cbind, lapply(sgn$signalPlus, function(x) {
                sum(x[rngPlus])
            })))
        mcols(rngMinus) = as.matrix(
            do.call(cbind, lapply(sgn$signalMinus, function(x) {
                sum(x[rngMinus])
            })))
        # sort ranges
        rngCov = c(rngPlus, rngMinus)
        rngCov = GenomeInfoDb::sortSeqlevels(rngCov)
        rngCov = sort(rngCov)
        # manage return type
        returnType = match.arg(returnType,
                                choices = c("GRanges", "matrix", "data.frame"))
        if (returnType == "GRanges") {
            retObj = rngCov
        }
        if (returnType == "matrix") {
            retObj = as.matrix(mcols(rngCov))
        }
        if (returnType == "data.frame") {
            retObj = as.data.frame(mcols(rngCov))
        }
    }
    return(retObj)
}

# Make Matrix from background Signal
#------------------------
makeBsBackgroundMatrix <- function(geneRanges, object, offset, matchBy = "geneID"){
```

```
    # reassign input
    genes = geneRanges
    bs = getRanges(object)
    bsWidth = unique(width(bs))
    signal = getSignal(object)

        # prepare the background region
        gen = genes[genes$geneID %in% bs$geneID]
        gen = sort(sortSeqlevels(gen))
        gen = as(gen, "GRangesList")
        # group binding sites per gene
        bs = sort(sortSeqlevels(bs))
        bsNames = mcols(bs) %>%
            as.data.frame() %>%
            group_by(geneID) %>%
            mutate(name = paste0("bs",seq_along(geneID))) %>%
            pull(name)
        names(bs) = bsNames
        bs = split(bs, bs$geneID)
        idx = match(names(gen), names(bs))
        bs = bs[idx]

    # add a protective range around each binding site
    # bsOffset = bs + offset
    bsOffset = resize(bs, fix = "center", width = bsWidth + offset)

    # remove binding sites ranges from gene ranges to create background
    bac = GenomicRanges::disjoin(pc(gen, bsOffset), with.revmap = TRUE)
    bac = unlist(bac)
    len = lapply(bac$revmap, length)
    bac = bac[len == 1]
    bac = split(bac, names(bac))
    bac = unlist(bac, use.names = F)

    # count crosslinks in background regions summarized by gene
    bac = coverageBySignal(range = bac, signal = signal, returnType = "GRanges")
    colnames(mcols(bac)) = paste0("counts.bg.", colnames(mcols(bac)))
    mcols(bac)$geneID = sapply(strsplit(names(bac),"\\."), `[`, 1)
    # mcols(bac)$geneID = names(bac)
    bacCounts = as.data.frame(mcols(bac))
    bacCounts = bacCounts %>%
        group_by(geneID) %>%
        summarize_if(is.numeric, sum) %>%
        as.data.frame()

    # count crosslinks in binding sites
    bs = unlist(bs)
    bsInitial = bs
    bsCounts = coverageBySignal(range = bs, signal = signal, returnType = "GRanges")
    colnames(mcols(bsCounts)) = paste0("counts.bs.", colnames(mcols(bsCounts)))

    # set matching IDs
    bsCounts$geneID = sapply(strsplit(names(bsCounts),"\\."), `[`, 1)
```

```
    bsCounts$PeakID = names(bsCounts)
    bsCounts = as.data.frame(mcols(bsCounts))

    # match peak and gene counts
    idx = match(bsCounts$geneID, bacCounts$geneID)
    comb = cbind.data.frame(bsCounts, bacCounts[idx,])
    comb$geneID = NULL
    comb$PeakID = NULL

    # combine counts and peak ranges for output
    idx = match(names(bs), rownames(comb))
    mcols(bs) = comb[idx,]

    # remove duplicated binding sites
    if (any(duplicated(bs))) {
        message(paste0("Found ", length(duplicated(bs)[duplicated(bs) == TRUE]),
                       " duplicated ranges. These are removed. "))
        bsDub = bs[duplicated(bs)]
        bs = bs[! bs %in% bsDub]
    }


    # match binding site counts and existing meta data
    idx = match(names(bs), names(bsInitial))
    mcols(bs) = cbind(mcols(bsInitial[idx]), mcols(bs))

    return(bs)
}


countObj = makeBsBackgroundMatrix(object = bdsFinal, geneRanges = gns, offset = 5, matchBy = "geneID")
```

Differential analysis of binding sites

```
### ============================================================================
### Run test with DESeq
### ----------------------------------------------------------------------------
#
count_matrix = as.data.frame(mcols(countObj))
count_matrix = count_matrix %>% dplyr::select(starts_with("counts"))

# internal modification for col data
colDataMod = rbind.data.frame(colData,colData)
colDataMod$type = c(rep("bs", nrow(colData)),
                    rep("bg", nrow(colData)))
colDataMod$condition = factor(colDataMod$condition, levels = c("KO", "WT"))
colDataMod$type = factor(colDataMod$type, levels = c("bs", "bg"))

# create SE object
se = SummarizedExperiment(assays = list(counts = as.matrix(count_matrix)),
                          rowRanges = granges(countObj), colData = colDataMod)

# set design - YOUs design
```

```
ddsBs = DESeqDataSet(se, design = ~condition + type + condition:type)
ddsBs$condition = relevel(ddsBs$condition, "WT")
ddsBs$type = relevel(ddsBs$type, "bg")
ddsBs = DESeq(ddsBs, test="LRT", reduced = ~ condition + type)
resBs = results(ddsBs, name = "conditionKO.typebs") # needs relevel of type to bg
resShrinkBs = lfcShrink(ddsBs, res = resBs, coef = "conditionKO.typebs", type = "ashr")


# deseq fetch results
diff_bs = countObj
colnames(resBs) = paste0("resBs.", colnames(resBs))
idx = match(names(diff_bs), rownames(resBs))
mcols(diff_bs) = cbind(mcols(diff_bs), resBs[idx,])
```

Differential analysis of background

```
### ============================================================================
### Run Background test with DESeq
### ----------------------------------------------------------------------------
#
# construct background signal dataframe to test for the background level change
count_matrix_bg = as.data.frame(mcols(countObj)) %>%
    dplyr::select(starts_with("counts.bg")) %>%
    unique()
#rownames(count_matrix_bg) = sapply(strsplit(rownames(count_matrix_bg),"\\."), `[`, 1)

# create SE object
seBg = SummarizedExperiment(assays = list(counts = as.matrix(count_matrix_bg)), colData = colData)

# DESeq design
ddsBg = DESeqDataSet(seBg, design = ~ condition)
ddsBg$condition = relevel(ddsBg$condition, "WT")
ddsBg = DESeq(ddsBg)
resBg = results(ddsBg, contrast = c("condition", "KO", "WT"))
resShrinkBg = lfcShrink(ddsBg, res = resBg, contrast = c("condition", "KO", "WT"), type = "ashr")

# add results to final differential object
colnames(resBg) = paste0("resBg.", colnames(resBg))
idx = match(names(diff_bs), rownames(resBg))
mcols(diff_bs) = cbind(mcols(diff_bs), resBg[idx,])
```

Initial results

```
### ============================================================================
### Numbers overview table
### ----------------------------------------------------------------------------
###
df = data.frame(Name = c("Tested", "Not Significant", "Significant", "Sig+Up", "Sig+Down"),
                N = c(nrow(resBs),
                      nrow(subset(resBs, resBs.padj >= 0.05)),
                      nrow(subset(resBs, resBs.padj < 0.05)),
                      nrow(subset(resBs, resBs.padj < 0.05 & resBs.log2FoldChange > 0)),
```

Table 2: Results by numbers. Significant based on IHW adjusted P value threshold 0.05.

Name	Count (#N)	Percentage (%)
Tested	28,253	100.00
Not Significant	27,115	95.97
Significant	1,138	4.03
Sig+Up	296	26.01
Sig+Down	842	73.99

```
                      nrow(subset(resBs, resBs.padj < 0.05 & resBs.log2FoldChange < 0))
                      ))
df$Per = c(100,
           nrow(subset(resBs, resBs.padj >= 0.05)) / nrow(resBs) * 100,
           nrow(subset(resBs, resBs.padj < 0.05)) / nrow(resBs) * 100,
           nrow(subset(resBs, resBs.padj < 0.05 & resBs.log2FoldChange > 0)) / nrow(subset(resBs, resBs
           nrow(subset(resBs, resBs.padj < 0.05 & resBs.log2FoldChange < 0)) / nrow(subset(resBs, resBs
           )
df$Per = round(df$Per, digits = 2)
colnames(df) = c("Name", "Count (#N)", "Percentage (%)")

kable(myNumberFormat(df), caption = "Results by numbers. Significant based on IHW adjusted P value thres
```

Gene expression filter

Some genes might not be expressed in both conditions. Binding sites on a gene that is for example only expressed in the *WT* but not in the *KO* condition will all appear to be down regulated. In reality we can not tell if such sites changed, because the hosting gene is not expressed. To prevent these effects I filtered all genes with a merged binding site (so the combined set from both conditions) before running the differential binding search. As cutoff a TPM > 50 is used.

```
# cutoff based on gene

counts = counts(ddsBg, normalized = FALSE)
# Extract all exons of a gen
exonsByGene <- exonsBy(annoDb, by = "gene")
# reduce overlapping exons to a single region
reducedExonsByGene <- GenomicRanges::reduce(exonsByGene)
# Approximate the gene length as the sum of the its exons lengths
geneLengths <- sum(width(reducedExonsByGene))
# Adapt gene IDs
names(geneLengths) = sapply(strsplit(names(geneLengths),"\\."), `[`, 1)
# Re-order the vector of gene lengths to match the order in the counts
counts_genes <- sapply(strsplit(rownames(counts),"\\."),  `[`, 1)
geneLengths <- geneLengths[match(counts_genes, names(geneLengths))]
# First step1 in TPM calculation
tpms <- counts / geneLengths
# Second step in TPM calculation
tpms <- t(t(tpms) * 1e6 / colSums(tpms, na.rm = T)) %>% as.data.frame()
# # Adapt gene IDs
rownames(tpms) = sapply(strsplit(rownames(tpms),"\\."), `[`, 1)
colnames(tpms) = paste0("tpm.", colnames(tpms) )

# add tpm to final object
```

```
idx <- match(diff_bs$geneID, rownames(tpms))
mcols(diff_bs) <- cbind(mcols(diff_bs), tpms[idx,])
```

A gene is considered expressed by a condition if at least two of the three replicates have a TPM over 50. Each gene must be found expressed in both conditions to be considered for the differential binding analysis.

```
# filter for tpm cutoff in 2 samples per condition
diff_bs$BS_ID <- names(diff_bs)
diff_bs <- as.data.frame(diff_bs) %>%
  rowwise() %>%
  mutate(tpm_support_KO = sum(c((tpm.counts.bg.1_KO > tpm_cut),
                                (tpm.counts.bg.2_KO > tpm_cut),
                                (tpm.counts.bg.3_KO > tpm_cut))),
         tpm_support_WT = sum(c((tpm.counts.bg.4_WT > tpm_cut),
                                (tpm.counts.bg.5_WT > tpm_cut),
                                (tpm.counts.bg.6_WT > tpm_cut))),
         tpm_supported = (tpm_support_KO > 1) & (tpm_support_WT > 1))


# subset by cutoff
diff_bs <- diff_bs %>% subset(tpm_supported )
```

The expression filtering resulted in 3,779 target genes, with 26,462 binding sites.

Results (after Gene Expression filter)

```
### ============================================================================
### Numbers overview table
### ----------------------------------------------------------------------------
###
df = data.frame(Name = c("Tested", "Not Significant", "Significant", "Sig+Up", "Sig+Down"),
                N = c(nrow(diff_bs),
                      nrow(subset(diff_bs, resBs.padj >= 0.05)),
                      nrow(subset(diff_bs, resBs.padj < 0.05)),
                      nrow(subset(diff_bs, resBs.padj < 0.05 & resBs.log2FoldChange > 0)),
                      nrow(subset(diff_bs, resBs.padj < 0.05 & resBs.log2FoldChange < 0))
                      ))
df$Per = c(100,
           nrow(subset(diff_bs, resBs.padj >= 0.05)) / nrow(diff_bs) * 100,
           nrow(subset(diff_bs, resBs.padj < 0.05)) / nrow(diff_bs) * 100,
           nrow(subset(diff_bs, resBs.padj < 0.05 & resBs.log2FoldChange > 0)) / nrow(subset(diff_bs, r
           nrow(subset(diff_bs, resBs.padj < 0.05 & resBs.log2FoldChange < 0)) / nrow(subset(diff_bs, r
           )
df$Per = round(df$Per, digits = 2)
colnames(df) = c("Name", "Count (#N)", "Percentage (%)")

kable(myNumberFormat(df), caption = "Results by numbers. Significant based on IHW adjusted P value thres
```

```
d = lapply(seq(0, 1, by = 0.05), function(cutoff){
    diff_bs %>%
        dplyr::select(region, resBs.log2FoldChange, resBs.padj) %>%
        rename("lfc" = "resBs.log2FoldChange" ,  "padj" = "resBs.padj" ) %>%
        base::subset(padj <= cutoff) %>%
        mutate(dir = ifelse(lfc > 0, "Up", "Down")) %>%
```

Table 3: Results by numbers. Significant based on IHW adjusted P value threshold 0.05.

Name	Count (#N)	Percentage (%)
Tested	26,462	100.00
Not Significant	25,388	95.94
Significant	1,074	4.06
Sig+Up	278	25.88
Sig+Down	796	74.12

```
        dplyr::select(region, dir) %>%
        mutate(cutoff = cutoff)
})
d = do.call("rbind", d)


df2 = d %>%
    subset(., cutoff == 0.05) %>%
    subset(., !is.na(region)) %>%
    subset(., region != "outside")%>%
    group_by(region, dir) %>%
    tally() %>%
    ungroup() %>%
    mutate(sum = sum(n)) %>%
    mutate(percentage = round(n/sum, digits = 4)*100)
df2 = df2 %>%
    mutate(region = factor(region, levels = c("utr5", "intron", "cds", "utr3"))) %>%
    arrange(region)

p1 = ggplot(df2, aes(x = dir, y = percentage, fill = region)) +
    geom_col(position = "stack") +
    theme_pub() +
    scale_fill_npg() +
    labs(
        title = "Significantly regulated binding sites per region and direction",
        x = "Direction of regulation",
        y = "Percentage",
        fill = "Region"
    ) +
    coord_flip() +
    theme(legend.position = "top")


p1
```

```
ggsave(p1, filename = paste0(out, "/S1j_differential_regionwise.pdf"), height = 6, width = 6, units = "
```

We cut of the p-value at < 0.05 and use no cutoff on the foldchange.

Final numbers

```
diff_bs_sig <- diff_bs %>% subset(resBs.padj < 0.05)
```

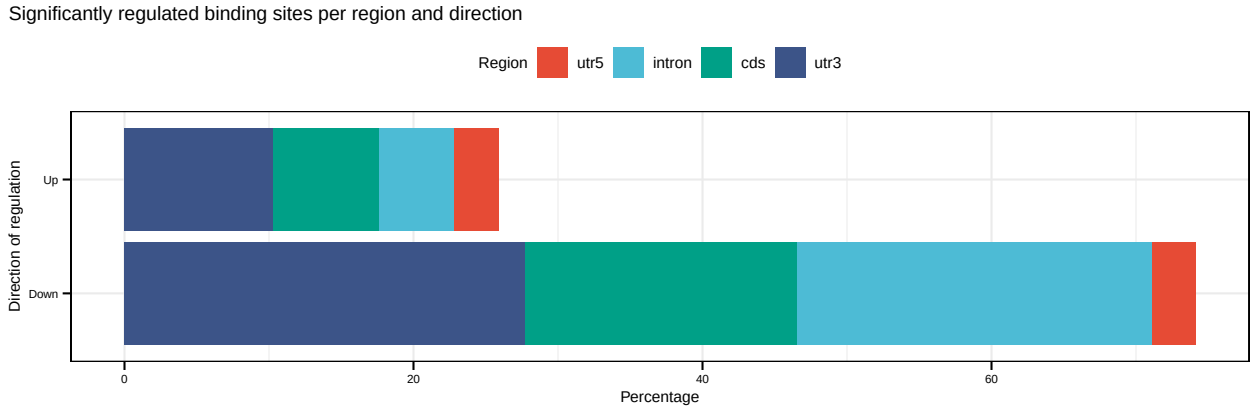

Figure 1: Significantly regulated binding sites per region and direction. The percentage is calculated from all significant binding sites at adusted P value cutoff of 0.05.

```
table(diff_bs_sig$region)
```

```
##
##      cds   intron outside     utr3     utr5
##      268      306        1      391       63
```

```
sum(table(diff_bs_sig$region))
```

```
## [1] 1029
```

```
sum(is.na(diff_bs_sig$region))
```

```
## [1] 45
```

Final results vulcano

```
# MA + Volcano plots
df = diff_bs %>%
    mutate(sig = (ifelse(resBs.padj < 0.05 , TRUE, FALSE))) %>%
    mutate(dir = factor(ifelse(resBs.log2FoldChange > 0,
                               "Up",
                               "Down"),
                        level = c("Up", "Down"))) %>%
    mutate(sigDir = ifelse(sig == TRUE & dir == "Up",
                           "Up",
                           ifelse(sig == TRUE & dir == "Down",
                                  "Down",
                                  "Not"))) %>%
    mutate(sigDir = factor(sigDir,
                           levels = c("Not", "Up", "Down"))) %>%
    arrange(sigDir)


pnv = ggplot(df, aes(x = resBs.log2FoldChange, y = -log10(resBs.padj), fill = factor(sigDir, levels = c
    geom_point(shape = 21, stroke = 0.5, size = 3, color = "black") +
  scale_fill_manual(values = c(farbeneg, "#e3a09cff",  farbe3))+
  geom_point(data=df[df$geneName == "Zfp36l1" & df$sig == TRUE,], aes(x = resBs.log2FoldChange, y = -log
```

```
  geom_text_repel(data=df[df$geneName == "Zfp3611" & df$sig == TRUE,], aes(x = resBs.log2FoldChange, y =
    geom_vline(xintercept = 0, color = "black", alpha = .5) +
    theme_paper()+
    theme(legend.key.size = unit(1, 'cm'), legend.position = "top") +
    labs(
        x = "Fold-change (log2)",
        y = "Adjusted P value (-log10)",
        color = "Regulation",
        fill = "Regulation")
pnv
```

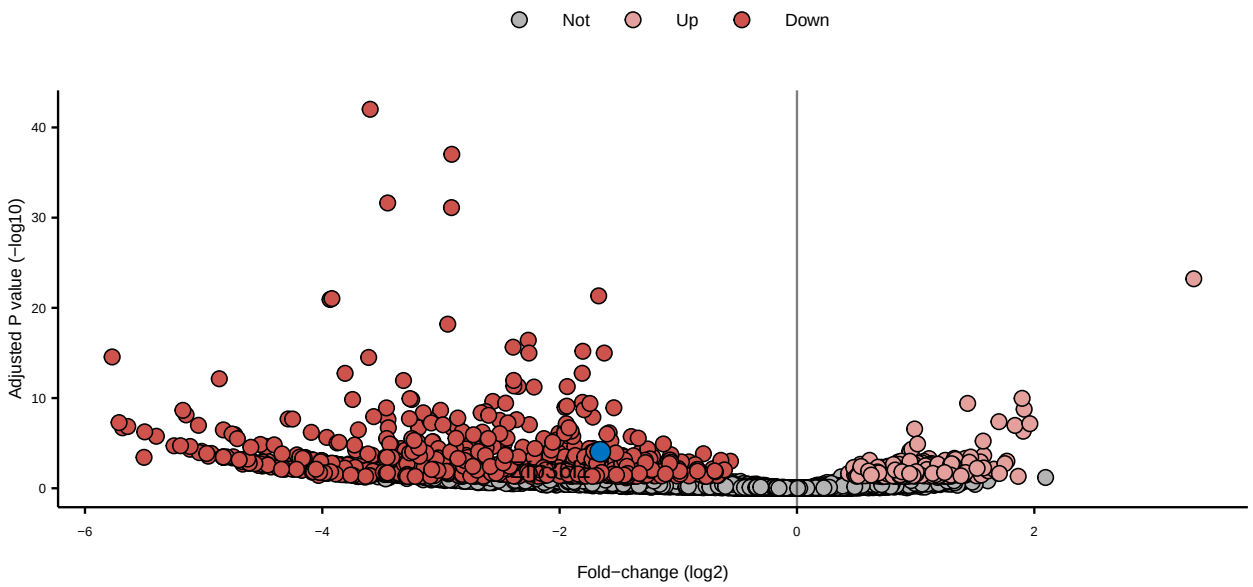

Figure 2: MA and Volcano plots. With adjusted P value and fold-change cutoffs. Custom annotaitons - significant only.

```
ggsave(pnv, file= paste0(out, "Figure2C_Differntial_binding_vulcano.pdf"), height = 7, width = 6, units
```

Export results

```
# export results
saveRDS(diff_bs, file = paste0(out, "BsDifferentialResult.rds"))

write.csv(diff_bs, paste0(out, "Supplementary_table2_differentail_binding.csv"))
```

Session Info

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## 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] IHW_1.26.0                   DESeq2_1.38.3
##  [3] SummarizedExperiment_1.28.0 MatrixGenerics_1.10.0
##  [5] matrixStats_1.0.0           ggrastr_1.0.2
##  [7] ggsci_3.0.0                 ggpointdensity_0.1.0
##  [9] tidyr_1.3.0                 tibble_3.2.1
## [11] patchwork_1.1.2             ggtext_0.1.2
## [13] forcats_1.0.0               ComplexHeatmap_2.14.0
## [15] BindingSiteFinder_1.4.0     viridis_0.6.3
## [17] viridisLite_0.4.2           gridExtra_2.3
## [19] ggrepel_0.9.3               kableExtra_1.3.4
## [21] GenomicFeatures_1.50.4      UpSetR_1.4.0
## [23] reshape2_1.4.4              dplyr_1.1.2
## [25] AnnotationDbi_1.60.2        Biobase_2.58.0
## [27] ggplot2_3.4.2               rtracklayer_1.58.0
## [29] GenomicRanges_1.50.2        GenomeInfoDb_1.34.9
## [31] IRanges_2.32.0              S4Vectors_0.36.2
## [33] BiocGenerics_0.44.0         knitr_1.43
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.3               tidyselect_1.2.0         RSQLite_2.3.1
##   [4] htmlwidgets_1.6.2        BiocParallel_1.32.6      munsell_0.5.0
##   [7] codetools_0.2-19         ragg_1.2.5               interp_1.1-4
##  [10] withr_2.5.0              colorspace_2.1-0         filelock_1.0.2
##  [13] highr_0.10               rstudioapi_0.15.0        ggsignif_0.6.4
##  [16] labeling_0.4.2           slam_0.1-50              GenomeInfoDbData_1.2.9
##  [19] lpsymphony_1.26.3        mixsqp_0.3-48            polyclip_1.10-4
##  [22] bit64_4.0.5              farver_2.1.1             rprojroot_2.0.3
##  [25] vctrs_0.6.3              generics_0.1.3           xfun_0.39
##  [28] biovizBase_1.46.0        BiocFileCache_2.6.1      R6_2.5.1
##  [31] doParallel_1.0.17        ggbeeswarm_0.7.2         clue_0.3-64
##  [34] invgamma_1.1             locfit_1.5-9.8           AnnotationFilter_1.22.0
##  [37] bitops_1.0-7             cachem_1.0.8             DelayedArray_0.24.0
##  [40] BiocIO_1.8.0             scales_1.2.1             nnet_7.3-19
##  [43] beeswarm_0.4.0           gtable_0.3.3             ensembldb_2.22.0
##  [46] rlang_1.1.1              systemfonts_1.0.4        GlobalOptions_0.1.2
##  [49] rstatix_0.7.2            lazyeval_0.2.2           dichromat_2.0-0.1
##  [52] broom_1.0.5              checkmate_2.2.0          abind_1.4-5
##  [55] yaml_2.3.7               backports_1.4.1          Hmisc_5.1-0
##  [58] gridtext_0.1.5           tools_4.2.2              RColorBrewer_1.1-3
##  [61] Rcpp_1.0.11              plyr_1.8.8               base64enc_0.1-3
##  [64] progress_1.2.2           zlibbioc_1.44.0          purrr_1.0.1
##  [67] RCurl_1.98-1.12          prettyunits_1.1.1        ggpubr_0.6.0
##  [70] rpart_4.1.19             deldir_1.0-9             GetoptLong_1.0.5
##  [73] ashr_2.2-54              cluster_2.1.4            here_1.0.1
```

```
##  [76] magrittr_2.0.3          data.table_1.14.8        circlize_0.4.15
##  [79] truncnorm_1.0-9         SQUAREM_2021.1           ProtGenerics_1.30.0
##  [82] hms_1.1.3               evaluate_0.21            xtable_1.8-4
##  [85] XML_3.99-0.14           jpeg_0.1-10              shape_1.4.6
##  [88] compiler_4.2.2          biomaRt_2.54.1           crayon_1.5.2
##  [91] htmltools_0.5.5         Formula_1.2-5            geneplotter_1.76.0
##  [94] DBI_1.1.3               tweenr_2.0.2             dbplyr_2.3.3
##  [97] MASS_7.3-60             rappdirs_0.3.3           car_3.1-2
## [100] Matrix_1.5-4.1          cli_3.6.1                parallel_4.2.2
## [103] Gviz_1.42.1             pkgconfig_2.0.3          GenomicAlignments_1.34.1
## [106] foreign_0.8-84          xml2_1.3.5               foreach_1.5.2
## [109] svglite_2.1.1           annotate_1.76.0          vipor_0.4.5
## [112] webshot_0.5.5           XVector_0.38.0           rvest_1.0.3
## [115] stringr_1.5.0           VariantAnnotation_1.44.1 digest_0.6.33
## [118] Biostrings_2.66.0       rmarkdown_2.23           htmlTable_2.4.1
## [121] restfulr_0.0.15         curl_5.0.1               Rsamtools_2.14.0
## [124] rjson_0.2.21            lifecycle_1.0.3          carData_3.0-5
## [127] BSgenome_1.66.3         fansi_1.0.4              pillar_1.9.0
## [130] lattice_0.21-8          KEGGREST_1.38.0          fastmap_1.1.1
## [133] httr_1.4.6              glue_1.6.2               fdrtool_1.2.17
## [136] png_0.1-8               iterators_1.0.14         bit_4.0.5
## [139] ggforce_0.4.1           stringi_1.7.12           blob_1.2.4
## [142] textshaping_0.3.6       latticeExtra_0.6-30      memoise_2.0.1
## [145] irlba_2.3.5.1
```