

# Enriched binding sites definition

Melina Klostermann

03 April, 2023

## Contents

1	Libraries and settings	1
2	Functions	2
3	What was done?	2
4	Files	3
5	Obtained pureclip sites (from peak calling)	4
6	Compute binding sites	5
7	Downstream characterization	25
8	Save final binding sites	28
9	Session Info	28

## 1 Libraries and settings

```
# -----  
# libraries  
# -----  
  
library(rtracklayer)  
library(GenomicRanges)  
library(ggplot2)  
library(dplyr)  
library(reshape2)  
library(GenomicFeatures)  
library(BindingSiteFinder)  
library(ComplexHeatmap)  
library(tibble)  
library(tidyr)  
library(dplyr)  
library(ggrepel)  
library(ggpointdensity)  
library(purrr)  
library(Biostrings)  
library(BSgenome.Mmusculus.UCSC.mm10)
```

```

library(knitr)
library(kableExtra)

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

out <- "/Users/melinaklostermann/Documents/projects/AgoCLIP_miR181/R_github/miR181_paper/Methods/mir181"

```

## 2 Functions

```

# -----
# utility functions
# -----

basicVectorToNiceDf <- function(x){
  # NOTE MK you could do all starting from the 3. line in one mutate command, might be faster and easier
  df = data.frame(Type = names(table(x$olType)), Freq = as.vector(table(x$olType)))
  df = df[order(df$Freq, decreasing = F),]
  df$Type = factor(df$Type, levels = df$Type)
  df$Frac = df$Freq / sum(df$Freq)
  df$ymax = cumsum(df$Frac)
  df$ymin = c(0, head(df$ymax, n=-1))
  df$labPos = (df$ymax + df$ymin) / 2
  df$NFrac = round(df$Frac * 100)
  df$NFrac2 = round(df$Freq / sum(df$Freq), digits = 4)
  df$NFracNice = df$NFrac2 * 100
  df$labelNice = paste0(format(df$Freq, big.mark = ",", decimal.mark = "."), " (", df$NFracNice, "%)")
  df$labelNice2 = paste0(df$Type, ": ", format(df$Freq, big.mark = ",", decimal.mark = "."), " (", df$NFracNice, "%)")
  return(df)
}

```

## 3 What was done?

- Here I define mir181 binding sites on basis of the enriched mir181 miR-eCLIP data.
- I merge the bam files and bw files from the non-chimeric and chimeric (after trimming off the mir part) read from the enriched mir181 data.
- Peaks are called with pureclip on the .bam merge of all three replicates.
- Then binding sites are defined with BindingSiteFinder. The used binding site size is 7nt to match with the AGO binding site definition. The xxx plots show that 7nt fits the binding site patterns well.
- Binding sites are filtered for their reproducibility in at least 2 of 3 samples.
- Then bindingsites are matched to the bound gene and the bound gene region.

*NOTE:* Large parts of code and text are from the BindingSiteFinder Vignette. For a detailed explanation see <https://www.bioconductor.org/packages/release/bioc/vignettes/BindingSiteFinder/inst/doc/vignette.html>

## 4 Files

### 4.1 Merge crosslinks of chimeric and non-chimeric reads

Here we use a merge of the chimeric and non-chimeric crosslinks from the enriched mir181 data. .bw files from chimeric and non-chimeric reads which are outputted from Raccon are merged in the commandline with uscs-bigWigMerge.

```
# -----  
# Merge bw non-chimeric and chimeric sample wise  
# -----  
  
bigWigMerge crosslinks_chimeric_sampleX.bw crosslinks_non-chimeric_sampleX.bw crosslinks_mixed_sampleX.bw  
  
LC_COLLATE=C sort -k1,1 -k2,2n -k3,3n crosslinks_mixed_sampleX.bed -o crosslinks_mixed_sampleX.sort.bed  
  
bedGraphToBigWig crosslinks_mixed_sampleX.sort.bed GRCm38.p6.genome.fa.fai crosslinks_mixed_sampleX.sort.bw
```

### 4.2 Merge bam files run pureclip

We use the peakcaller pureclip to detect crosslink peaks. pureclip is run on the merge of the three samples with both chimeirc and non-chimeric crosslinks.

```
# -----  
# Merge bam non-chimeric and chimeric (all samples together)  
# -----  
  
samtools merge -o crosslinks_mixed_all_samples.bam crosslinks_chimeric_all_samples.bamm crosslinks_non-chimeric_all_samples.bam  
samtools sort crosslinks_mixed_all_samples.bam crosslinks_mixed_all_samples.sort.bam  
samtools index crosslinks_mixed_all_samples.sort.bam  
  
# -----  
# Run pureclip  
# -----  
  
pureclip \  
-i crosslinks_mixed_all_samples.sort.bam\  
-bai crosslinks_mixed_all_samples.sort.bam.bai \  
-g GRCm38.p6.genome.strict.IUPAC.fa \  
-nt 10 \  
-o mixed_IP_WT_miR181_pureclip_sites_230323.bed \  
  
# -----  
# -----  
# Files  
# -----  
# -----  
  
# -----  
# annotation  
# -----  
  
annoDb <- loadDb("/Users/melinaklostermann/Documents/projects/AgoCLIP_miR181/R_github/miR181_paper/Method/annoDb.sqlite")  
gns <- readRDS("/Users/melinaklostermann/Documents/projects/AgoCLIP_miR181/R_github/miR181_paper/Method/gns.rds")  
  
# -----
```

seqnames	start	end	width	strand	score
chr1	6240152	6240152	1	+	0.976837
chr1	6240153	6240153	1	+	0.563029
chr1	6244795	6244795	1	+	0.461608
chr1	6244796	6244796	1	+	0.607505
chr1	6248280	6248280	1	+	1.300870
chr1	6248281	6248281	1	+	0.878214

```
# pureclip sites (from peak calling)
# -----
pureclip_sites <- "/Users/melinaklostermann/Documents/projects/AgoCLIP_miR181/pureclip/mixed_IP_WT_miR181"

# -----
# crosslinks
# -----

# crosslinks mixed chimeric, non-chimeric
clipFiles = "/Users/melinaklostermann/Documents/projects/AgoCLIP_miR181/pipe_output_22_02_14/chimeric/c"
clipFiles = list.files(clipFiles, pattern = ".bw$", full.names = TRUE)
clipFiles = clipFiles[!grepl("Inp", clipFiles)]
clipFiles = clipFiles[grepl("miR181_", clipFiles)]
clipFiles = clipFiles[grepl("WT", clipFiles)]
clipFilesP = clipFiles[grepl("plus", clipFiles)]
clipFilesM = clipFiles[grepl("minus", clipFiles)]
```

## 5 Obtained pureclip sites (from peak calling)

```
# -----
# Get and clean pureclip sites
# -----

pureclip_sites <- import.bedGraph(pureclip_sites)
pureclip_sites = as.data.frame(pureclip_sites)
pureclip_sites$score = pureclip_sites$NA.
pureclip_sites$strand = pureclip_sites$NA..1
pureclip_sites$NA. = NULL
pureclip_sites$NA..1 = NULL
pureclip_sites$NA..2 = NULL
pureclip_sites = makeGRangesFromDataFrame(pureclip_sites, keep.extra.columns = T)
pureclip_sites = keepStandardChromosomes(pureclip_sites, pruning.mode = "coarse")

kable(head(pureclip_sites)) %>%
  kable_material(c("striped", "hover"))
```

→ Number of pureclip sites: 33930

## 6 Compute binding sites

### 6.1 Settings for binding sites

```
# -----  
# set BS final options  
# -----  
bsSize_Final = 7  
minWidth_Final = 2  
minCrosslinks_Final = 2  
minClSites_Final = 2
```

### 6.2 Make BindingSiteFinder object

```
# -----  
# Organize clip data in dataframe for binding site finder  
# -----  
  
colData= data.frame(  
  id = c(1:3),  
  condition = factor(c("WT", "WT", "WT"),  
                     levels = c("WT")),  
  clPlus = clipFilesP ,  
  clMinus = clipFilesM)  
  
# Make BindingSiteFinder object  
bds = BSFDataSetFromBigWig(ranges = pureclip_sites, meta = colData)  
bds  
  
## Object of class BSFDataSet  
## Contained ranges: 33.930  
## ----> Number of chromosomes: 22  
## ----> Ranges width: 1  
## Contained Signal: 5,156,221  
## Contained conditions: WT
```

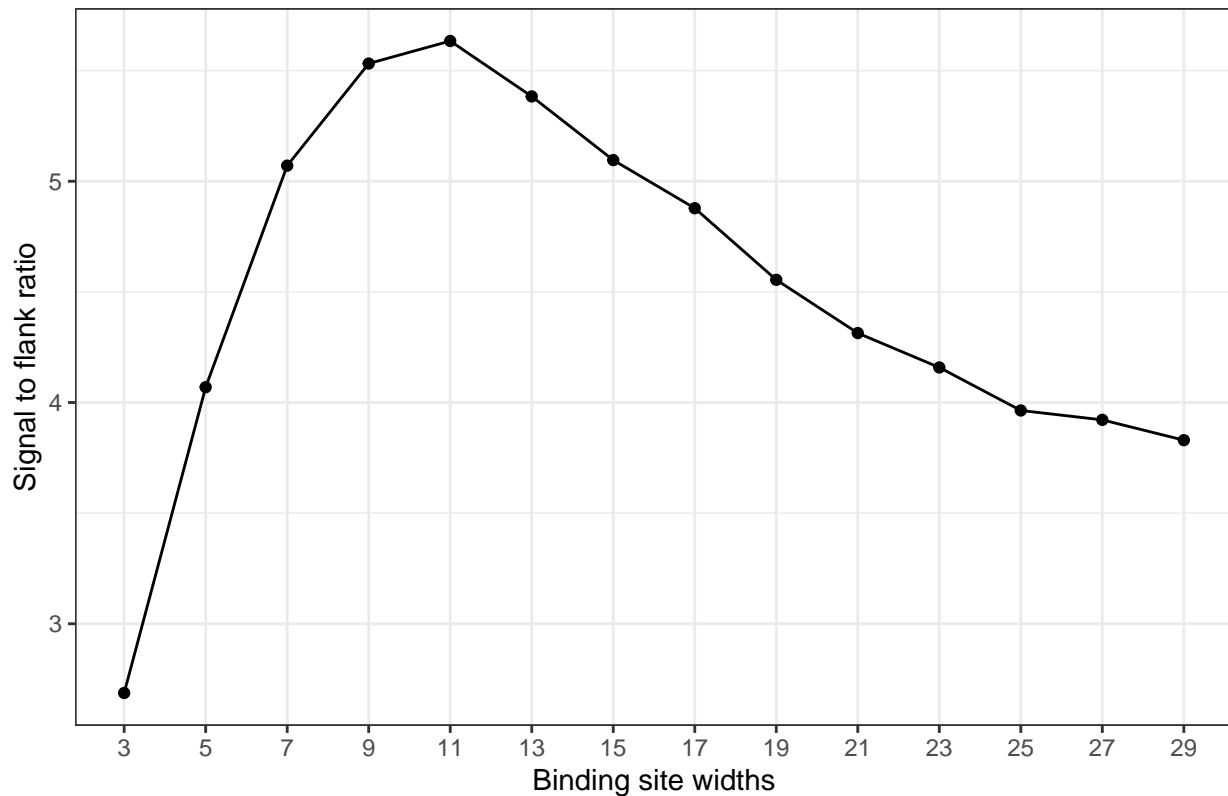
### 6.3 Controls for binding site width setting

We use a binding site width of 7nt because this setting was chosen for the AGO binding sites. With the following plots we doublecheck that 7 is a suitable size for the enriched mir181 binding sites.

The optimal binding site width is determined by the binding site signal to noise ratio. For each binding site, the ratio of crosslink events within the binding site and of crosslink events flanking the binding site to both sides is determined.

```
# -----  
# SupportRatio plot  
# -----  
supportRatioPlot(bds, bsWidths = seq(from = 3, to = 29, by = 2), minWidth = minWidth_Final, minClSites =
```

### Binding site widths signal ratios



This plot shows that the optimal resolution of signal in binding sites would be for 9nt or 11nt binding sites. For further visual inspection selected binding site width are plotted as count profiles centered around the binding sites central position.

```
# -----
# RangeCoverage plot
# -----

bds1 <- makeBindingSites(object = bds, bsSize = 3, minWidth = 2,
                          minCrosslinks = 2, minClSites = 2, sub.chr = "chr1")

bds2 <- makeBindingSites(object = bds, bsSize = 5, minWidth = 2,
                          minCrosslinks = 2, minClSites = 2, sub.chr = "chr1")

bds3 <- makeBindingSites(object = bds, bsSize = 7, minWidth = 2,
                          minCrosslinks = 2, minClSites = 2, sub.chr = "chr1")

bds4 <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
                          minCrosslinks = 2, minClSites = 2, sub.chr = "chr1")

bds5 <- makeBindingSites(object = bds, bsSize = 15, minWidth = 2,
                          minCrosslinks = 2, minClSites = 2, sub.chr = "chr1")

bds6 <- makeBindingSites(object = bds, bsSize = 17, minWidth = 2,
                          minCrosslinks = 2, minClSites = 2, sub.chr = "chr1")

bds7 <- makeBindingSites(object = bds, bsSize = 11, minWidth = 2,
```

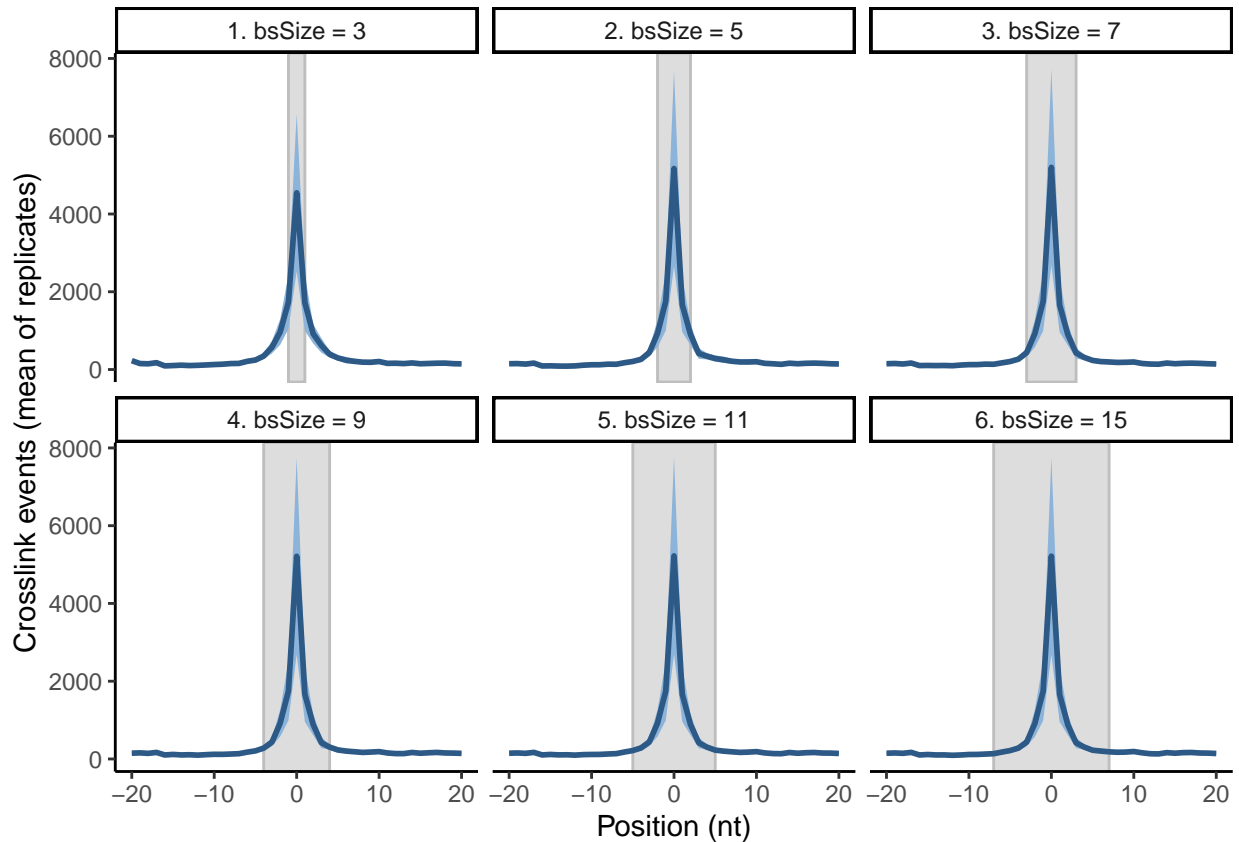
```

minCrosslinks = 2, minClSites = 2, sub.chr = "chr1")

l = list(`1. bsSize = 3` = bds1, `2. bsSize = 5` = bds2, `3. bsSize = 7` = bds3,
        `4. bsSize = 9` = bds4, `5. bsSize = 11` = bds7, `6. bsSize = 15` = bds5)

rangeCoveragePlot(l, width = 20)

```



In generally, a binding site width of 9nt would be ideal for this data. However, for reasons of comparability we stay with a size of 7nt.

## 6.4 Make binding sites

```

# -----
# Make binding sites
# -----
bds_sites <- makeBindingSites(object = bds, bsSize = bsSize_Final, minWidth = minWidth_Final,
                             minCrosslinks = minCrosslinks_Final, minClSites = minClSites_Final)

bds_sites

## Object of class BSFDataSet
## Contained ranges: 7.809
## ----> Number of chromosomes: 22
## ----> Ranges width: 7
## Contained Signal: 5,156,221
## Contained conditions: WT

```

Option	nRanges
inputRanges	33930
mergeCrosslinkSites	8729
minCrosslinks	8084
minCISites	7930
centerIsCISite	7925
centerIsSummit	7809

```
# make GrangesObject of binding sites
```

```
bds_sites_gr = getRanges(bds_sites)
```

```
bds_sites_gr
```

```
## GRanges object with 7809 ranges and 0 metadata columns:
```

```
##      seqnames      ranges strand
```

```
##      <Rle>        <IRanges> <Rle>
```

```
##      1      chr1 6240149-6240155      +
```

```
##      2      chr1 6244793-6244799      +
```

```
##      3      chr1 6248277-6248283      +
```

```
##      4      chr1 6248959-6248965      +
```

```
##      5      chr1 6249310-6249316      +
```

```
##      ...      ...      ...      ...
```

```
## 8496      chrM    14296-14302      +
```

```
## 8497      chrM    15351-15357      +
```

```
## 8498      chrM      3842-3848      -
```

```
## 8499      chrM      5257-5263      -
```

```
## 8500      chrM    16189-16195      -
```

```
## -----
```

```
## seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

```
df = getSummary(bds_sites) # NOTE MK nice function
```

```
kable(df) %>%
```

```
  kable_material(c("striped", "hover"))
```

## 6.5 Binding site composition

Details about the binding site computation can be seen when counting the number of PureCLIP sites that make up a specific binding site.

```
# -----
```

```
# Check pureclip sites per binding site
```

```
# -----
```

```
df = countOverlaps(bds_sites_gr, pureclip_sites) %>%
```

```
  table() %>%
```

```
  as.data.frame() %>%
```

```
  rename("n" = ".")
```

```
ggplot(df, aes(x = n, y = Freq, fill = n)) +
```

```
  geom_col(color = "black") +
```

```
  scale_fill_grey() +
```

```
  theme_bw() +
```

```
  theme(legend.position = "none") +
```

```
  labs(
```

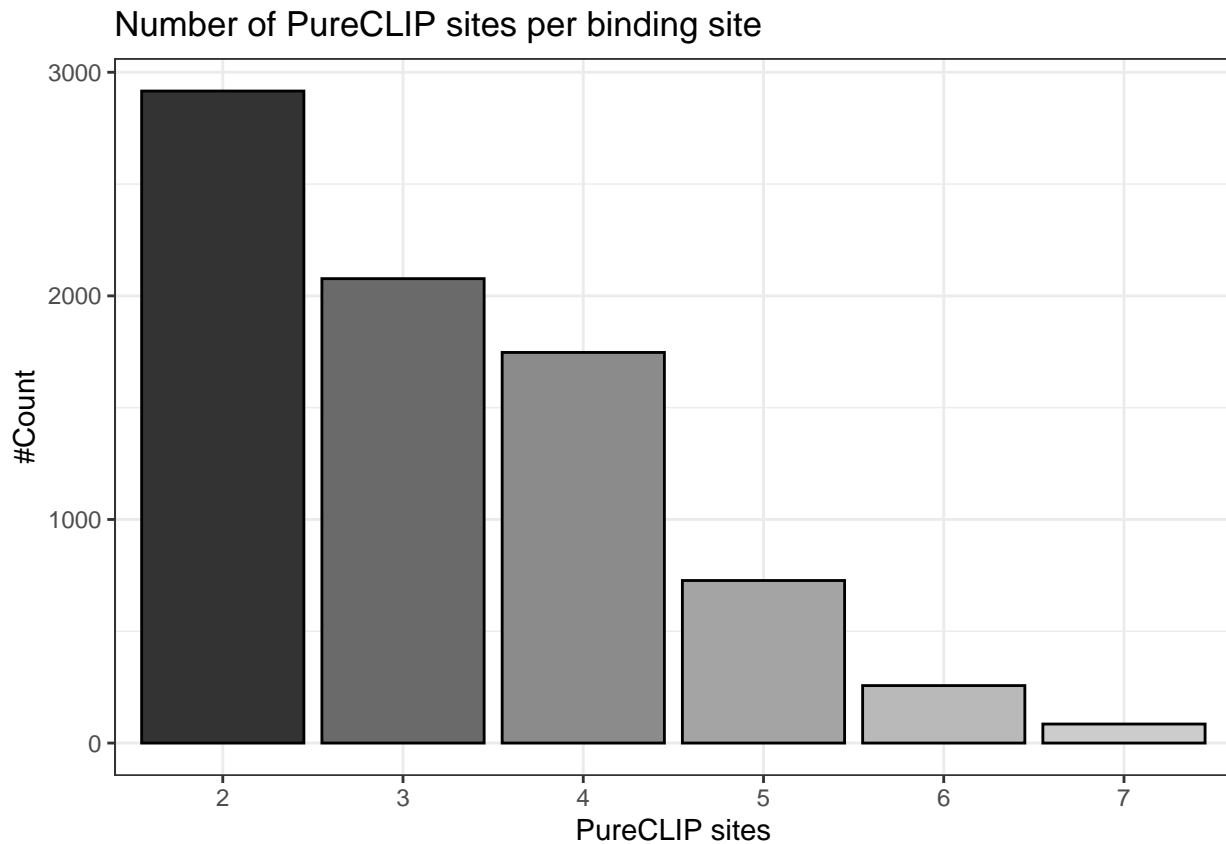
```
    title = "Number of PureCLIP sites per binding site",
```



```

x = "PureCLIP sites",
y = "#Count"
)

```



## 6.6 Reproducibility filter

Since peak calling is based on the merge of all replicates, we filter processed binding site for their support by the individual replicates. First, crosslinks are summed up per replicate creating a crosslink distribution. Then, a threshold is set for each replicate to the 5% quantile of that distribution. To account for low crosslink replicates, a lower boundary of 1 crosslink events per binding site is enforced.

```

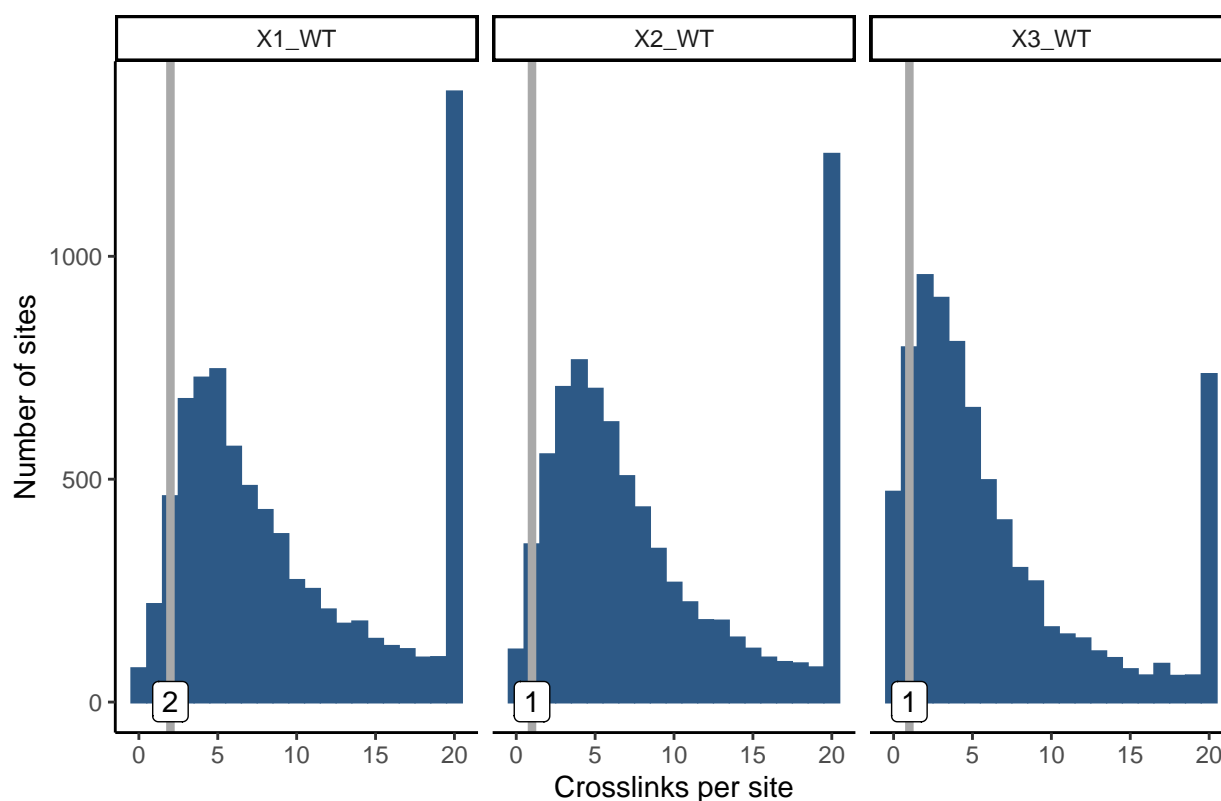
# -----
# Check reproducibility cutoff
# -----

p <- reproducibilityCutoffPlot(bds_sites, max.range = 20, cutoff = 0.05)

p

```

Cutoff: 5% WT; min.crosslinks = 1



Finally, a binding sites is deemed reproducible if the thresholds are met for all two of three replicates. The Upset plot below shows the support intersections of the three samples for the binding sites.

```
# -----
# Check reproducibility of binding sites between the three non-chimeric and the three chimeric samples
# -----
s1 = reproducibilityFilter(bds_sites, cutoff = 0.05, n.reps = 2, returnType = "data.frame")
m = make_comb_mat(s1, mode = "distinct")

df = strsplit(names(comb_degree(m)), "")
df = do.call(rbind, df)
df = as.matrix(df)
df = apply(df, 2, as.numeric)
df = as.data.frame(df)

df$support = rowSums(df)
rownames(df) = names(comb_degree(m))

df$status = ifelse(df$support == 3, "All",
                   ifelse(df$support == 2, "Two",
                           ifelse(df$support == 1, "One", "None")))

# plot Upset with Complex Heatmap package
ha = HeatmapAnnotation(
  col = list(select = c("All" = "#004d4d", "Two" = "aquamarine4", "One" = "aquamarine2", "None" = "grey"),
    "Sample intersections" = anno_barplot(comb_size(m), border = FALSE, gp = gpar(fill = "#595959"), height = 10))
```

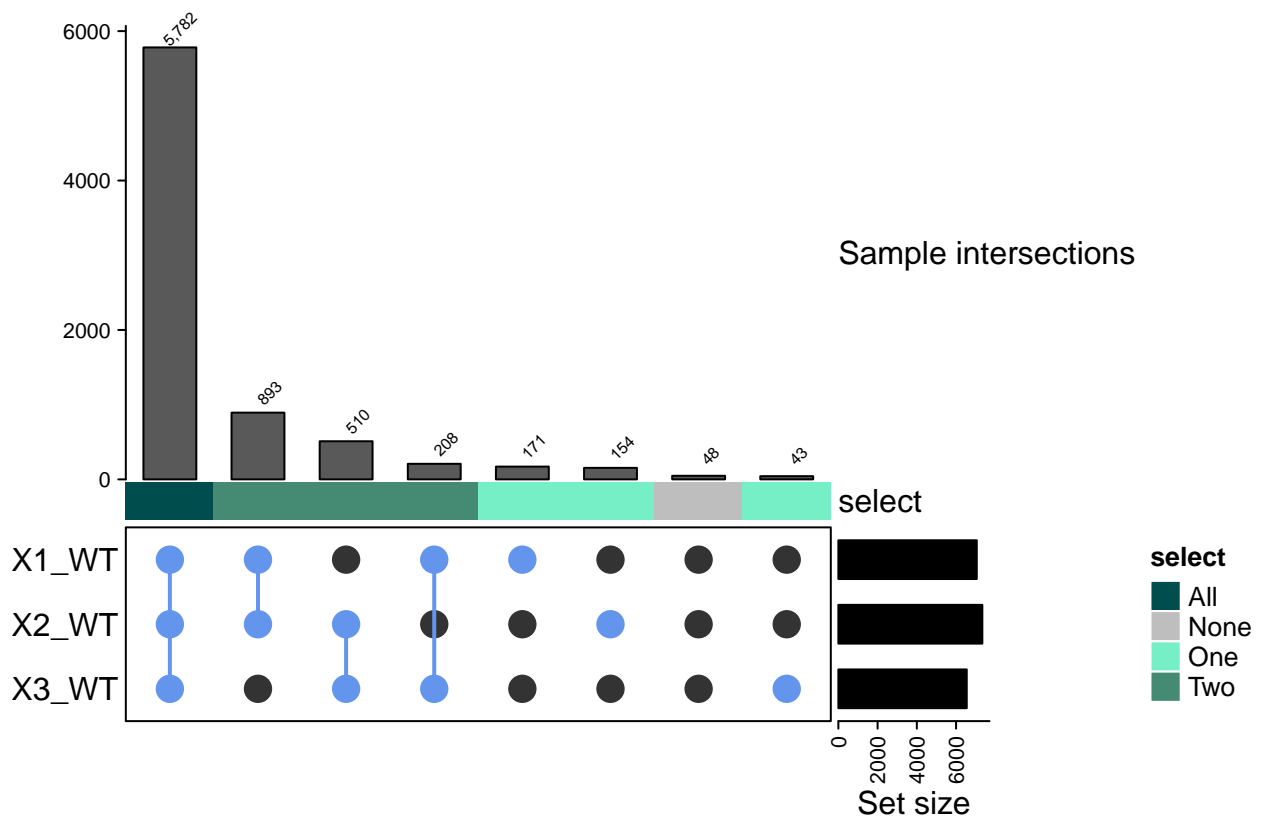
```

    select = df$status
)

ht = UpSet(m,
    set_order = colnames(s1),
    comb_order = order(comb_size(m), decreasing = T),
    top_annotation = ha,
    comb_col = "cornflowerblue", bg_col = "white", pt_size = unit(.5, "cm") ,
    border = T, lwd = 2, bg_pt_col = "#333333"
)

ss = set_size(m)
cs = comb_size(m)
ht = draw(ht, padding = unit(c(0, 0, 10, 0), "mm"))
od = column_order(ht)
decorate_annotation("Sample intersections", {
    grid.text(format(cs[od], big.mark = ",", decimal.mark = "."), x = seq_along(cs), y = unit(cs[od], "mm"),
        default.units = "native", just = c("left", "bottom"),
        gp = gpar(fontsize = 6, col = "black"), rot = 45)
})

```



## 6.7 Final binding sites

This leaves us with a final set of binding sites:

```

# -----
# Get reproducible binding sites
# -----

```

```

bdsFinal = reproducibilityFilter(bds_sites, cutoff = 0.05, n.reps = 2)
bdsFinal = annotateWithScore(bdsFinal, getRanges(bds))
bdsFinal_gr = getRanges(bdsFinal)

# kable(head(bdsFinal_gr)) %>%
#   kable_material(c("striped", "hover"))

```

We get 7393 reproducible binding sites.

## 6.8 Examples

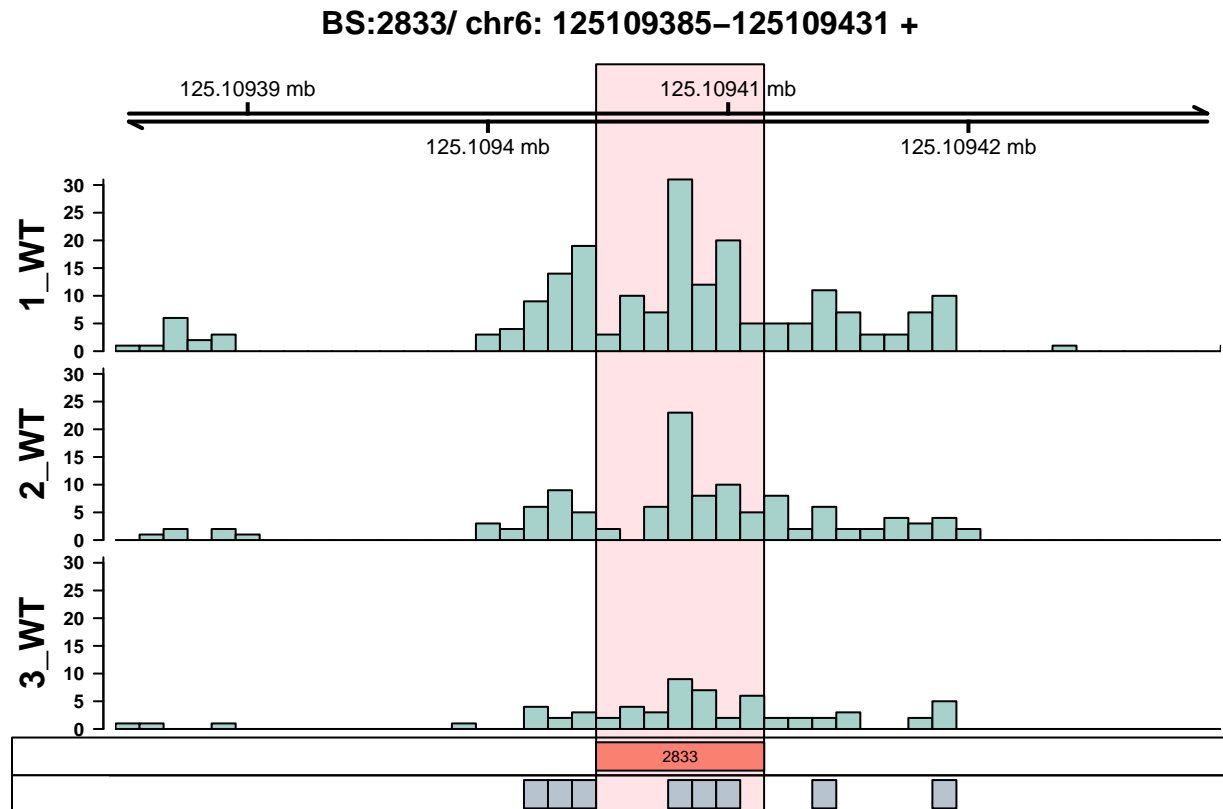
```

set.seed(123)

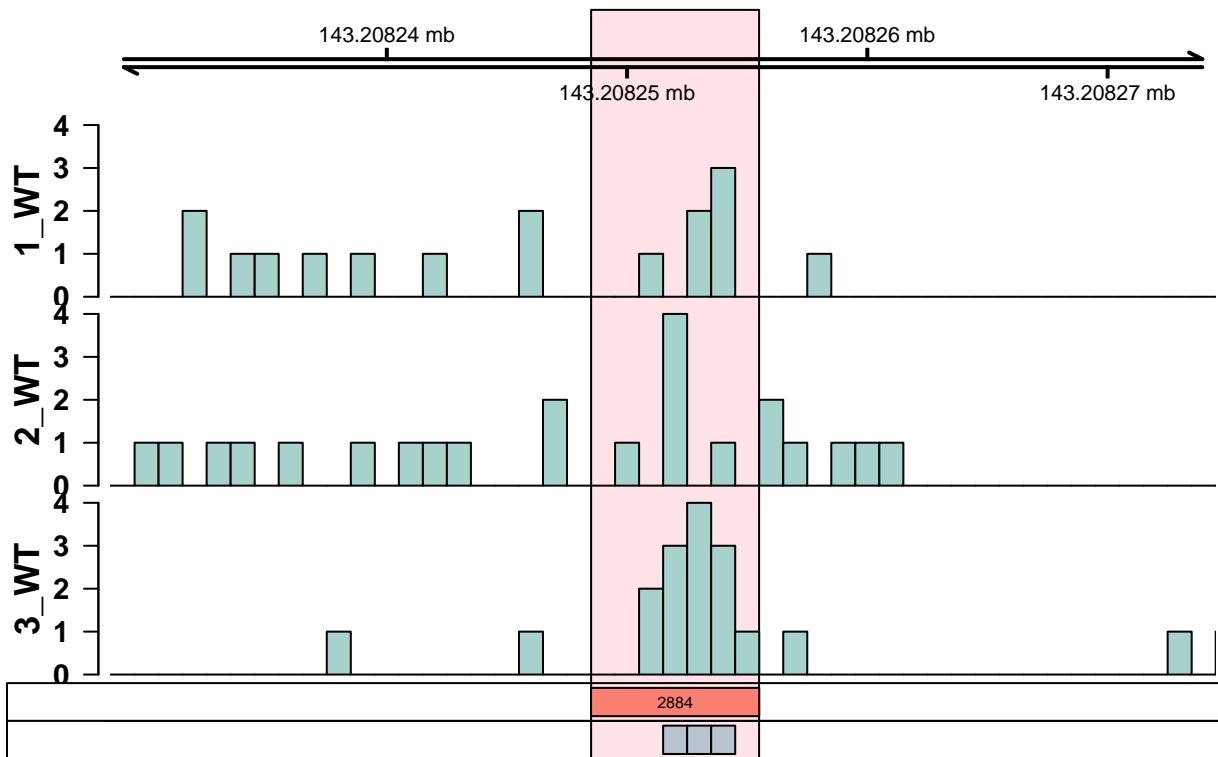
s <- sample(1:NROW(bdsFinal_gr), 8) %>% as.list(.)

lapply(s, function(x) bindingSiteCoveragePlot(bdsFinal,
                                              plotIdx = x,
                                              flankPos = 20,
                                              autoscale = TRUE,
                                              customRange = pureclip_sites,
                                              customRange.name = "pSites"))

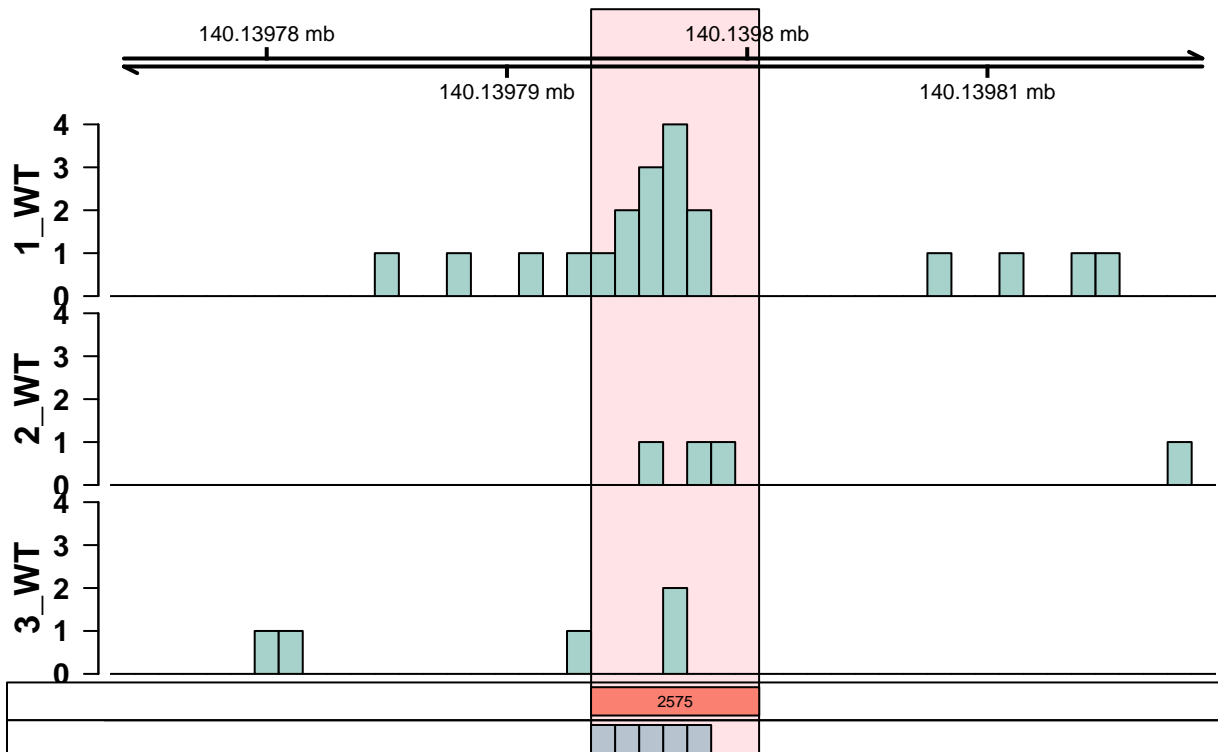
```



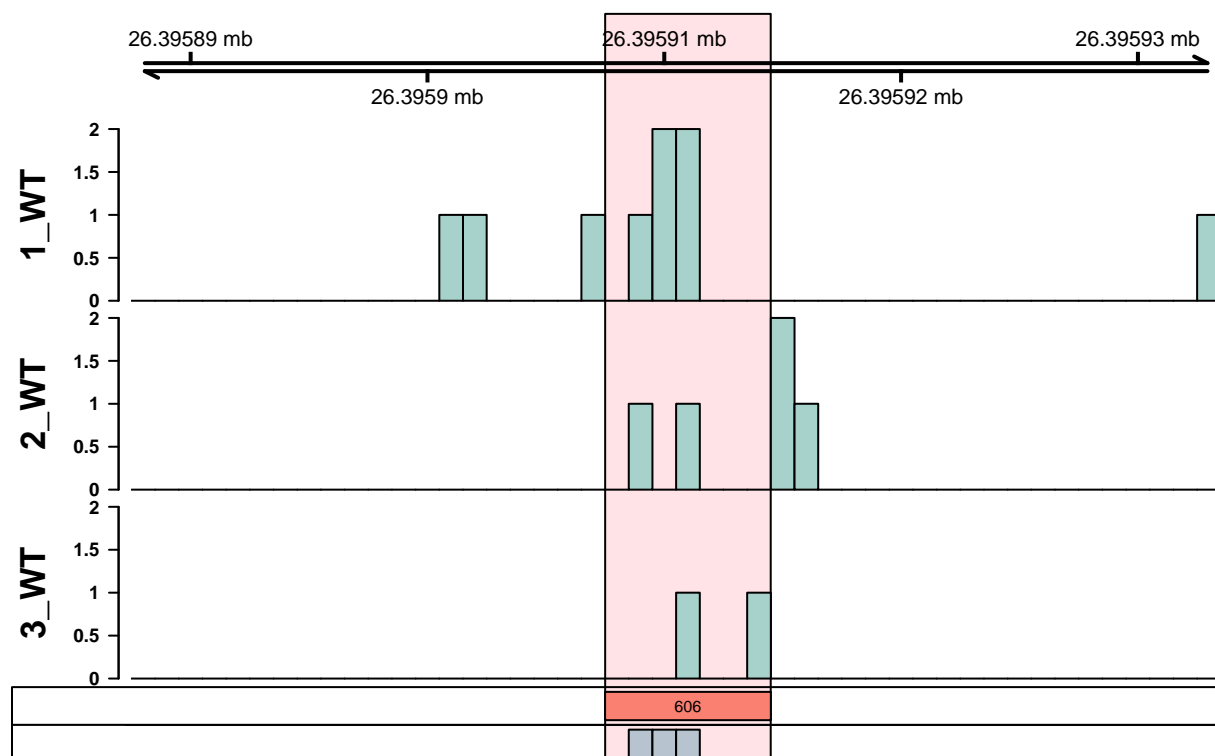
**BS:2884/ chr6: 143208229–143208275 +**



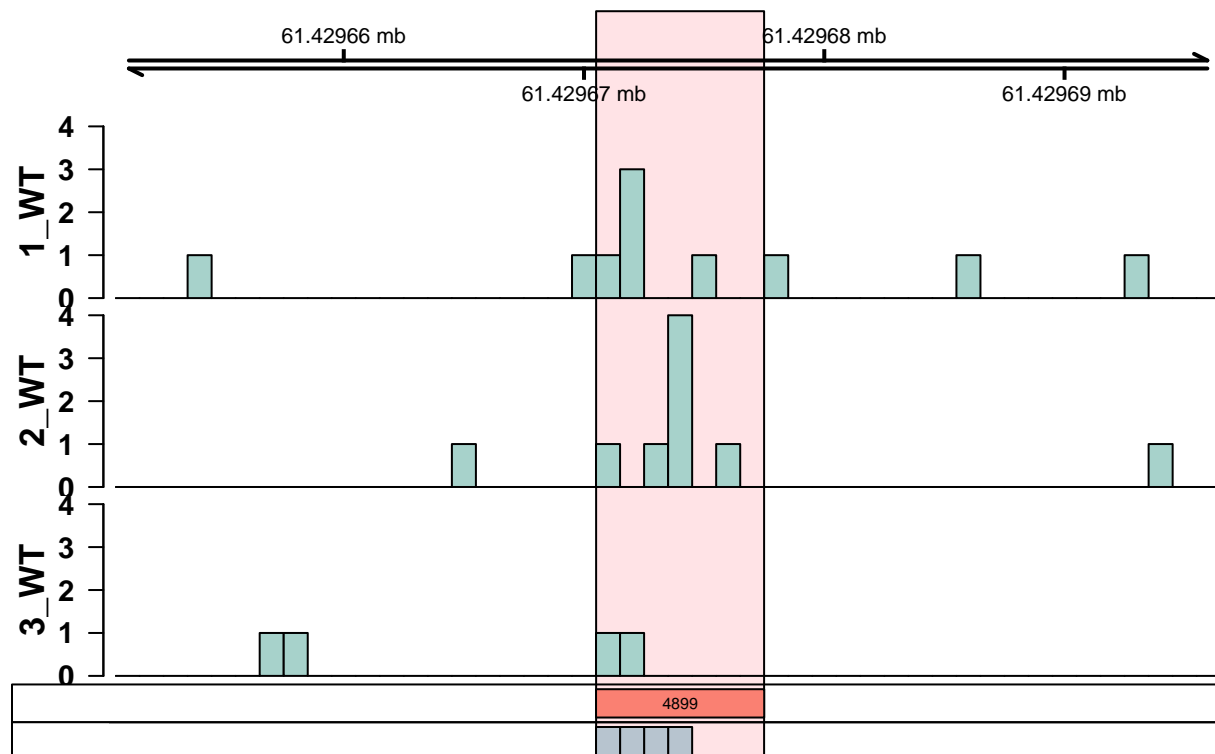
**BS:2575/ chr5: 140139774–140139820 -**



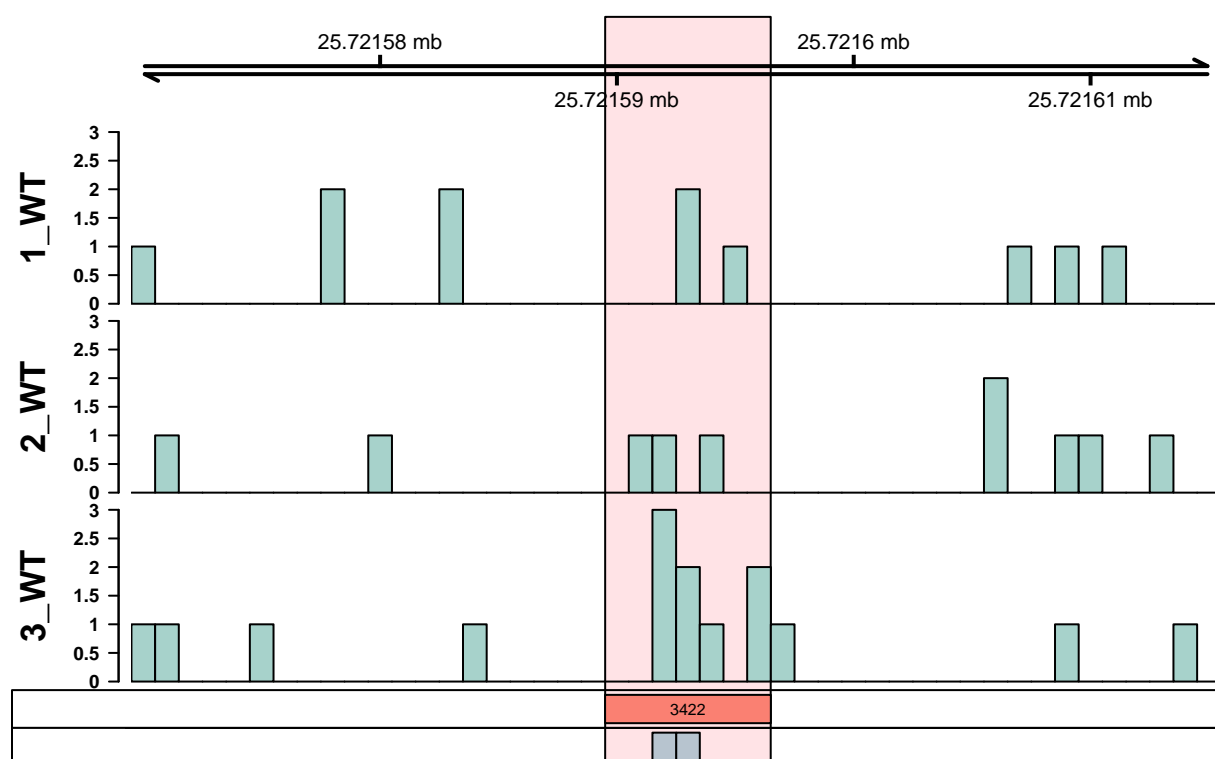
**BS:606/ chr2: 26395888–26395934 +**



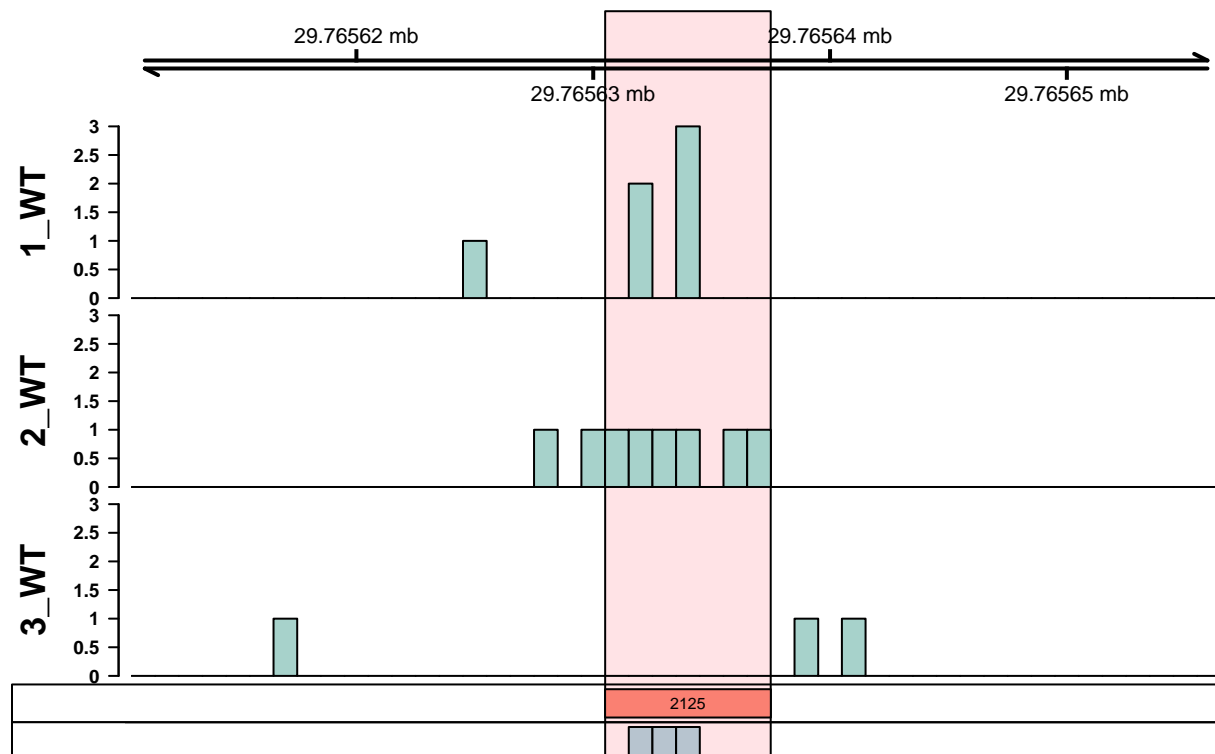
**BS:4899/ chr10: 61429651–61429697 –**



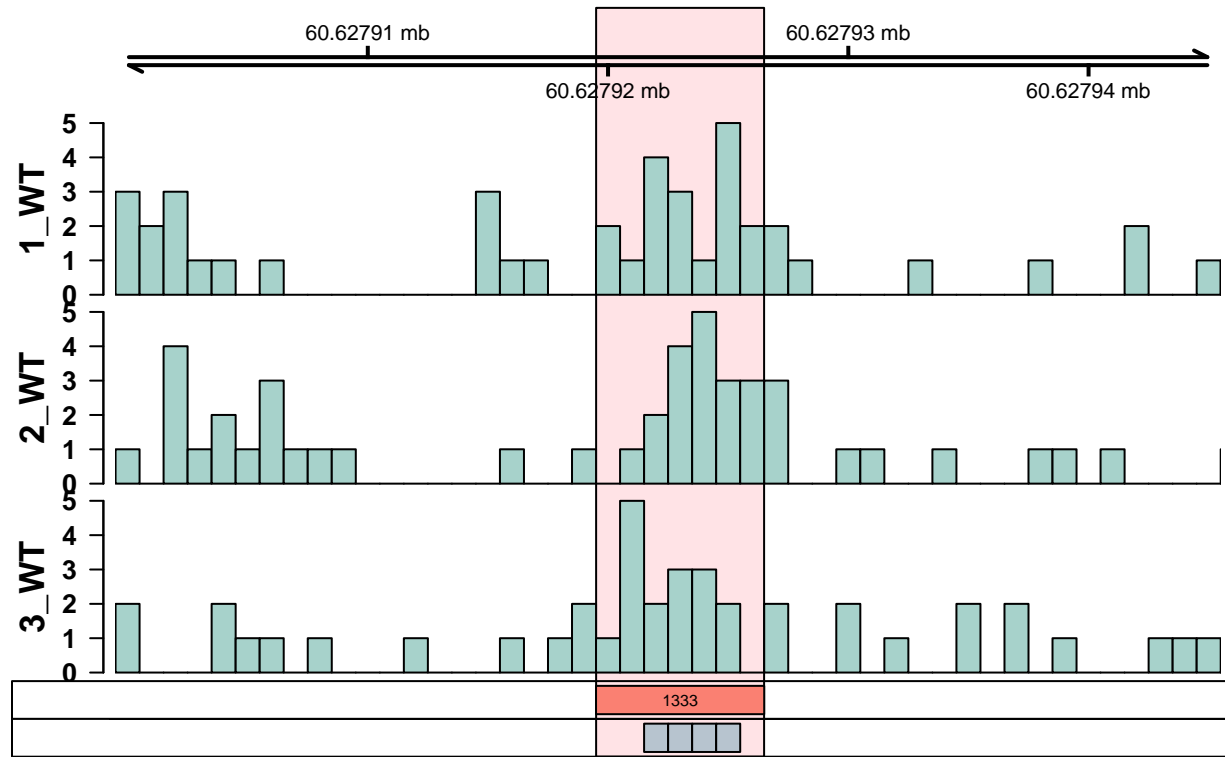
**BS:3422/ chr7: 25721570–25721616 –**



**BS:2125/ chr5: 29765611–29765657 +**



# BS:1333/ chr3: 60627900–60627946 +



```
## [[1]]
## [[1]]$Axis
## Genome axis 'Axis'
##
## [[1]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr6
## | positions: 47
## | samples:1
## | strand: +
##
## [[1]]$`2_WT`
## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr6
## | positions: 47
## | samples:1
## | strand: +
##
## [[1]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA
## | active chromosome: chr6
## | positions: 47
## | samples:1
## | strand: +
##
```



```

## [[1]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr6
## | annotation features: 1
##
## [[1]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr6
## | annotation features: 8
##
## [[1]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6 37.2000 42.07388 77.1672
## 1_WT      6 77.1672 42.07388 148.0004
## 2_WT      6 148.0004 42.07388 218.8336
## 3_WT      6 218.8336 42.07388 289.6667
## BS Other  6 289.6667 42.07388 303.8334
## pSites    6 303.8334 42.07388 318.0000
##
## Slot "tags":
## $title
##      Axis      1_WT      2_WT      3_WT  BS Other      pSites
##      "Axis"      "1_WT"      "2_WT"      "3_WT" "BS Other"      "pSites"
##
##
##
##
## [[2]]
## [[2]]$Axis
## Genome axis 'Axis'
##
## [[2]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr6
## | positions: 47
## | samples:1
## | strand: +
##
## [[2]]$`2_WT`
## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr6
## | positions: 47
## | samples:1
## | strand: +
##
## [[2]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA

```

```

## | active chromosome: chr6
## | positions: 47
## | samples:1
## | strand: +
##
## [[2]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr6
## | annotation features: 1
##
## [[2]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr6
## | annotation features: 3
##
## [[2]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6 37.2000 42.07388 77.1672
## 1_WT      6 77.1672 42.07388 148.0004
## 2_WT      6 148.0004 42.07388 218.8336
## 3_WT      6 218.8336 42.07388 289.6667
## BS Other  6 289.6667 42.07388 303.8334
## pSites    6 303.8334 42.07388 318.0000
##
## Slot "tags":
## $title
##      Axis      1_WT      2_WT      3_WT  BS Other      pSites
##      "Axis"      "1_WT"      "2_WT"      "3_WT" "BS Other"      "pSites"
##
##
##
##
## [[3]]
## [[3]]$Axis
## Genome axis 'Axis'
##
## [[3]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr5
## | positions: 47
## | samples:1
## | strand: -
##
## [[3]]$`2_WT`
## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr5
## | positions: 47
## | samples:1

```

```

## | strand: -
##
## [[3]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA
## | active chromosome: chr5
## | positions: 47
## | samples:1
## | strand: -
##
## [[3]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr5
## | annotation features: 1
##
## [[3]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr5
## | annotation features: 5
##
## [[3]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6  37.2000 42.07388  77.1672
## 1_WT      6  77.1672 42.07388 148.0004
## 2_WT      6 148.0004 42.07388 218.8336
## 3_WT      6 218.8336 42.07388 289.6667
## BS Other  6 289.6667 42.07388 303.8334
## pSites    6 303.8334 42.07388 318.0000
##
## Slot "tags":
## $title
##      Axis      1_WT      2_WT      3_WT  BS Other      pSites
##      "Axis"    "1_WT"    "2_WT"    "3_WT" "BS Other"  "pSites"
##
##
##
##
## [[4]]
## [[4]]$Axis
## Genome axis 'Axis'
##
## [[4]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr2
## | positions: 47
## | samples:1
## | strand: +
##
## [[4]]$`2_WT`

```

```

## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr2
## | positions: 47
## | samples:1
## | strand: +
##
## [[4]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA
## | active chromosome: chr2
## | positions: 47
## | samples:1
## | strand: +
##
## [[4]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr2
## | annotation features: 1
##
## [[4]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr2
## | annotation features: 3
##
## [[4]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6 37.2000 48.07868 77.1672
## 1_WT      6 77.1672 48.07868 148.0004
## 2_WT      6 148.0004 48.07868 218.8336
## 3_WT      6 218.8336 48.07868 289.6667
## BS Other  6 289.6667 48.07868 303.8334
## pSites    6 303.8334 48.07868 318.0000
##
## Slot "tags":
## $title
##      Axis      1_WT      2_WT      3_WT  BS Other      pSites
##      "Axis"    "1_WT"    "2_WT"    "3_WT" "BS Other"  "pSites"
##
##
##
##
## [[5]]
## [[5]]$Axis
## Genome axis 'Axis'
##
## [[5]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr10

```

```

## | positions: 47
## | samples:1
## | strand: -
##
## [[5]]$`2_WT`
## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr10
## | positions: 47
## | samples:1
## | strand: -
##
## [[5]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA
## | active chromosome: chr10
## | positions: 47
## | samples:1
## | strand: -
##
## [[5]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr10
## | annotation features: 1
##
## [[5]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr10
## | annotation features: 4
##
## [[5]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6  37.2000 42.07388 77.1672
## 1_WT      6  77.1672 42.07388 148.0004
## 2_WT      6 148.0004 42.07388 218.8336
## 3_WT      6 218.8336 42.07388 289.6667
## BS Other  6 289.6667 42.07388 303.8334
## pSites    6 303.8334 42.07388 318.0000
##
## Slot "tags":
## $title
##      Axis      1_WT      2_WT      3_WT  BS Other  pSites
##      "Axis"    "1_WT"    "2_WT"    "3_WT" "BS Other" "pSites"
##
##
##
##
## [[6]]
## [[6]]$Axis
## Genome axis 'Axis'

```

```

##
## [[6]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr7
## | positions: 47
## | samples:1
## | strand: -
##
## [[6]]$`2_WT`
## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr7
## | positions: 47
## | samples:1
## | strand: -
##
## [[6]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA
## | active chromosome: chr7
## | positions: 47
## | samples:1
## | strand: -
##
## [[6]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr7
## | annotation features: 1
##
## [[6]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr7
## | annotation features: 2
##
## [[6]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6  37.2000 48.07868 77.1672
## 1_WT      6  77.1672 48.07868 148.0004
## 2_WT      6 148.0004 48.07868 218.8336
## 3_WT      6 218.8336 48.07868 289.6667
## BS Other  6 289.6667 48.07868 303.8334
## pSites    6 303.8334 48.07868 318.0000
##
## Slot "tags":
## $title
##      Axis      1_WT      2_WT      3_WT  BS Other  pSites
##      "Axis"    "1_WT"    "2_WT"    "3_WT" "BS Other" "pSites"
##
##

```

```

##
##
## [[7]]
## [[7]]$Axis
## Genome axis 'Axis'
##
## [[7]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr5
## | positions: 47
## | samples:1
## | strand: +
##
## [[7]]$`2_WT`
## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr5
## | positions: 47
## | samples:1
## | strand: +
##
## [[7]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA
## | active chromosome: chr5
## | positions: 47
## | samples:1
## | strand: +
##
## [[7]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr5
## | annotation features: 1
##
## [[7]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr5
## | annotation features: 3
##
## [[7]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6  37.2000 48.07868 77.1672
## 1_WT      6  77.1672 48.07868 148.0004
## 2_WT      6 148.0004 48.07868 218.8336
## 3_WT      6 218.8336 48.07868 289.6667
## BS Other  6 289.6667 48.07868 303.8334
## pSites    6 303.8334 48.07868 318.0000
##
## Slot "tags":

```

```

## $title
##      Axis      1_WT      2_WT      3_WT  BS Other      pSites
##      "Axis"      "1_WT"      "2_WT"      "3_WT" "BS Other"      "pSites"
##
##
##
##
## [[8]]
## [[8]]$Axis
## Genome axis 'Axis'
##
## [[8]]$`1_WT`
## DataTrack '1_WT'
## | genome: NA
## | active chromosome: chr3
## | positions: 47
## | samples:1
## | strand: +
##
## [[8]]$`2_WT`
## DataTrack '2_WT'
## | genome: NA
## | active chromosome: chr3
## | positions: 47
## | samples:1
## | strand: +
##
## [[8]]$`3_WT`
## DataTrack '3_WT'
## | genome: NA
## | active chromosome: chr3
## | positions: 47
## | samples:1
## | strand: +
##
## [[8]]$`BS Other`
## GeneRegionTrack 'BS Other'
## | genome: NA
## | active chromosome: chr3
## | annotation features: 1
##
## [[8]]$pSites
## GeneRegionTrack 'pSites'
## | genome: NA
## | active chromosome: chr3
## | annotation features: 4
##
## [[8]]$titles
## An object of class "ImageMap"
## Slot "coords":
##      x1      y1      x2      y2
## Axis      6  37.2000 42.07388 77.1672
## 1_WT      6  77.1672 42.07388 148.0004
## 2_WT      6 148.0004 42.07388 218.8336

```



```
## 3_WT      6 218.8336 42.07388 289.6667
## BS Other  6 289.6667 42.07388 303.8334
## pSites    6 303.8334 42.07388 318.0000
##
## Slot "tags":
## $title
##      Axis      1_WT      2_WT      3_WT  BS Other      pSites
##      "Axis"      "1_WT"      "2_WT"      "3_WT" "BS Other"      "pSites"
```

## 7 Downstream characterization

Next, we assign each binding site to the hosting gene and transcript part, using the initially loaded gene annotation from GENCODE.

### 7.1 Assignment of binding sites to genes

Assigning each binding site to its hosting gene is done by computing the overlap of all binding sites with the gene annotation. This typically results in some binding sites overlapping multiple different genes.

```
#-----
#  gene type priority rule
#-----
selectTerms = c("protein_coding", "tRNA", "lincRNA", "miRNA", "snRNA")
rule = unique(gns$gene_type)
rule = rule[!rule %in% selectTerms]
rule = c(selectTerms, rule)

# filter out pseudo genes
gns <- gns[!grepl(gns$gene_type, pattern = "pseudo")]
```

To resolve these overlaps genes are assigned based on the following hierarchical order: “protein\_coding” > “tRNA” > “lincRNA” > “miRNA” > “snRNA”

```
#-----
#  Assign to genes
#-----

# get genes with binding signal
target_genes = subsetByOverlaps(gns, bdsFinal_gr)
df = findOverlaps(target_genes, bdsFinal_gr) %>% as.data.frame()

# split into easy and complex cases
idxDouble = df[duplicated(df$subjectHits),]
idxSingle = df[!duplicated(df$subjectHits),]

# handle single overlap cases
peaksRepoSingle = bdsFinal_gr[idxSingle$subjectHits]
mcols(peaksRepoSingle)$geneType = target_genes$gene_type[idxSingle$queryHits]
mcols(peaksRepoSingle)$geneName = target_genes$gene_name[idxSingle$queryHits]
mcols(peaksRepoSingle)$geneID = target_genes$gene_id[idxSingle$queryHits] %>% sub("\\..*", "", .)

# handle multi overlap cases
peaksRepoDouble = bdsFinal_gr[idxDouble$subjectHits]

peaksRepoDoubleCleaned = as(lapply(seq_along(peaksRepoDouble), function(x){
```

```

currPeak = peaksRepoDouble[x]
currTargets = subsetByOverlaps(target_genes, currPeak)
nOverlaps = length(currTargets)

# 1) take gene type as first criterion
# -> prefer the type that is first in the `rule` list

solution = unique(match(currTargets$gene_type, rule))
nSolutions = length(solution)

if (nSolutions == nOverlaps) {
  # solution successful
  mcols(currPeak)$geneType = currTargets$gene_type[min(solution)]
  mcols(currPeak)$geneName = currTargets$gene_name[min(solution)]
  mcols(currPeak)$geneID = currTargets$gene_id[min(solution)] %>% sub("\\..*", "", .)
}
if (nSolutions < nOverlaps) {
  # no solution found
  # -> Stop and return NA
  mcols(currPeak)$geneType = NA
  mcols(currPeak)$geneName = NA
  mcols(currPeak)$geneID = NA
}
return(currPeak)
}), "GRangesList")
peaksRepoDoubleCleaned = unlist(peaksRepoDoubleCleaned)
peaksRepoDoubleCleaned = peaksRepoDoubleCleaned[!is.na(peaksRepoDoubleCleaned$geneID)]

# assign peaks
bsGene = c(peaksRepoSingle, peaksRepoDoubleCleaned)
bsGene = sortSeqlevels(bsGene)
bsGene = sort(bsGene)
bsGene = unique(bsGene) # why is the unique necessary here?

# assign targets
target_genes = target_genes[target_genes$gene_id %in% bsGene$geneID]

```

We get `NROW(bsGene)` binding sites, that can be assigned to a gene. (Note: These are some less than the number of binding sites we had before, because, some binding sites might be outside of annotated genes.) The binding sites are positioned on `NROW(target_genes)` genes.

## 7.2 Assignment of binding sites to transcripts

```

### setting the hierarchical rule for assignment
rule = c("utr3", "utr5", "cds", "intron")

```

The transcript parts bound by the RBP are identified by overlapping the binding sites of protein-coding genes with the transcripts of these genes. The respective transcript region, such as intron, CDS or UTR can be deduced from these overlaps. Similar to the gene assignment, some binding sites might also overlap with multiple different annotated transcript parts. These are resolved by application of the hierarchical rule: utr3, utr5, cds, intron. Note that the majority vote system is not used here!

```

#-----
# Assignment of binding sites to transcripts

```

region	Freq
cds	1935
intron	1766
outside	4
utr3	2339
utr5	378

```
#-----
# this is only done for protein coding genes
targetsProt = target_genes[target_genes$gene_type == "protein_coding"]
bsProt = bsGene[bsGene$geneType == "protein_coding"]
bsNonCodeing = bsGene[bsGene$geneType != "protein_coding"]
mcols(bsNonCodeing)$region = NA

### count the overlap of each binding site within each part of the gene
# Count the overlaps of each binding site for each region of the transcript.

cdseq = cds(annoDb)
intrns = unlist(intronsByTranscript(annoDb))
utr3 = unlist(threeUTRsByTranscript(annoDb))
utr5 = unlist(fiveUTRsByTranscript(annoDb))
regions = list(CDS = cdseq, Intron = intrns, UTR3 = utr3, UTR5 = utr5)
# Count the overlaps of each binding site fore each region of the transcript.
cdseq = regions$CDS %>% countOverlaps(bsProt,.)
intrns = regions$Intron %>% countOverlaps(bsProt,.)
utr3 = regions$UTR3 %>% countOverlaps(bsProt,.)
utr5 = regions$UTR5 %>% countOverlaps(bsProt,.)
countDf = data.frame(cds = cdseq, intron = intrns, utr3 = utr3, utr5 = utr5)

# sort by rule
countDf = countDf[, rule] %>%
  as.matrix() %>%
  cbind.data.frame(., outside = ifelse(rowSums(countDf) == 0, 1, 0) )
names = colnames(countDf)

# disable majority vote -> set all counts to 1
countDf = as.matrix(countDf)
countDf[countDf > 0] = 1

region = apply(countDf, 1, function(x){ names[which.max(x)] })

# add region annotation to binding sites object
mcols(bsProt)$region = region

kable(table(region)) %>%
  kable_material(c("striped", "hover"))
```

## 8 Save final binding sites

```
bs_annotated <- c(bsProt, bsNonCodeing)

saveRDS(bs_annotated, paste0(out, Sys.Date(), "-BS_mir181_enriched.rds"))
```

## 9 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] kableExtra_1.3.4          BSgenome.Mmusculus.UCSC.mm10_1.4.3
## [3] BSgenome_1.66.2           Biostrings_2.66.0
## [5] XVector_0.38.0            purrr_1.0.1
## [7] ggpointdensity_0.1.0      ggrepel_0.9.2
## [9] tidyr_1.3.0               tibble_3.1.8
## [11] ComplexHeatmap_2.14.0     BindingSiteFinder_1.4.0
## [13] GenomicFeatures_1.50.4    AnnotationDbi_1.60.0
## [15] Biobase_2.58.0            reshape2_1.4.4
## [17] dplyr_1.0.10              ggplot2_3.4.0
## [19] rtracklayer_1.58.0        GenomicRanges_1.50.2
## [21] GenomeInfoDb_1.34.7       IRanges_2.32.0
## [23] S4Vectors_0.36.1         BiocGenerics_0.44.0
## [25] knitr_1.42
##
## loaded via a namespace (and not attached):
## [1] backports_1.4.1           circlize_0.4.15
## [3] Hmisc_4.7-2              systemfonts_1.0.4
## [5] BiocFileCache_2.6.0       plyr_1.8.8
## [7] lazyeval_0.2.2           splines_4.2.2
## [9] BiocParallel_1.32.5       digest_0.6.31
## [11] foreach_1.5.2            ensemblDb_2.22.0
## [13] htmltools_0.5.4          magick_2.7.3
## [15] fansi_1.0.4              magrittr_2.0.3
## [17] checkmate_2.1.0          memoise_2.0.1
## [19] cluster_2.1.4            doParallel_1.0.17
## [21] matrixStats_0.63.0       svglite_2.1.1
## [23] prettyunits_1.1.1        jpeg_0.1-10
```

## [25] colorspace_2.1-0	blob_1.2.3
## [27] rvest_1.0.3	rappdirs_0.3.3
## [29] xfun_0.36	crayon_1.5.2
## [31] RCurl_1.98-1.9	survival_3.5-0
## [33] VariantAnnotation_1.44.0	iterators_1.0.14
## [35] glue_1.6.2	polyclip_1.10-4
## [37] gtable_0.3.1	zlibbioc_1.44.0
## [39] webshot_0.5.4	GetoptLong_1.0.5
## [41] DelayedArray_0.24.0	shape_1.4.6
## [43] scales_1.2.1	DBI_1.1.3
## [45] Rcpp_1.0.10	viridisLite_0.4.1
## [47] progress_1.2.2	htmlTable_2.4.1
## [49] clue_0.3-63	foreign_0.8-84
## [51] bit_4.0.5	Formula_1.2-4
## [53] htmlwidgets_1.6.1	httr_1.4.4
## [55] RColorBrewer_1.1-3	ellipsis_0.3.2
## [57] pkgconfig_2.0.3	XML_3.99-0.13
## [59] farver_2.1.1	Gviz_1.42.0
## [61] nnet_7.3-18	dbplyr_2.3.0
## [63] deldir_1.0-6	utf8_1.2.2
## [65] labeling_0.4.2	tidyselect_1.2.0
## [67] rlang_1.0.6	munsell_0.5.0
## [69] tools_4.2.2	cachem_1.0.6
## [71] cli_3.6.0	generics_0.1.3
## [73] RSQLite_2.2.20	evaluate_0.20
## [75] stringr_1.5.0	fastmap_1.1.0
## [77] yaml_2.3.7	bit64_4.0.5
## [79] KEGGREST_1.38.0	AnnotationFilter_1.22.0
## [81] xml2_1.3.3	biomaRt_2.54.0
## [83] compiler_4.2.2	rstudioapi_0.14
## [85] filelock_1.0.2	curl_5.0.0
## [87] png_0.1-8	tweenr_2.0.2
## [89] stringi_1.7.12	highr_0.10
## [91] lattice_0.20-45	ProtGenerics_1.30.0
## [93] Matrix_1.5-3	vctrs_0.5.2
## [95] pillar_1.8.1	lifecycle_1.0.3
## [97] GlobalOptions_0.1.2	data.table_1.14.6
## [99] bitops_1.0-7	R6_2.5.1
## [101] BiocIO_1.8.0	latticeExtra_0.6-30
## [103] gridExtra_2.3	codetools_0.2-18
## [105] dichromat_2.0-0.1	MASS_7.3-58.2
## [107] assertthat_0.2.1	SummarizedExperiment_1.28.0
## [109] rjson_0.2.21	withr_2.5.0
## [111] GenomicAlignments_1.34.0	Rsamtools_2.14.0
## [113] GenomeInfoDbData_1.2.9	parallel_4.2.2
## [115] hms_1.1.2	rpart_4.1.19
## [117] rmarkdown_2.20	MatrixGenerics_1.10.0
## [119] Cairo_1.6-0	biovizBase_1.46.0
## [121] ggforce_0.4.1	base64enc_0.1-3
## [123] interp_1.1-3	restfulr_0.0.15