RNAhybrid_heatmaps_fig2

Nikita Verheyden

2023-04-12

Setup

directory

```
setwd("D:/Krueger_Lab/Publications/miR181_paper/Figure2/RNAhybrid")

set.seed(123)
```

packages

```
source("D:/Krueger_Lab/Publications/miR181_paper_v21022023/Figure_theme/theme_paper.R")
library(BSgenome.Mmusculus.UCSC.mm10)
```

```
## Loading required package: BSgenome

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
```

```
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomeInfoDb

## Loading required package: GenomicRanges

## Loading required package: Biostrings

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

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

## Loading required package: rtracklayer
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:Biostrings':
##
##     collapse, intersect, setdiff, setequal, union

## The following object is masked from 'package:XVector':
##
##     slice

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(seqinr)
```

```
##
## Attaching package: 'seqinr'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
## The following object is masked from 'package:Biostrings':
##
##     translate
```

```
library(circlize)
```

```
## ========================================
## circlize version 0.4.15
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
library(ComplexHeatmap)
```

```
## Loading required package: grid
```

```
##
## Attaching package: 'grid'
```

```
## The following object is masked from 'package:Biostrings':
##
##     pattern
```

```
## ========================================
## ComplexHeatmap version 2.15.2
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
```

```
## ========================================
```

Data

the files imported here are created with RNAhybrid with the "RNAhybrid fig 2" script

```
Personalized_Reader <- function(lambda){
 read.table(lambda, sep = ":") %>% select(V1, V5, V6, V7, V10, V11)}

#File lists
reslistA <- list.files(path = "D:/Krueger_Lab/Publications/miR181_paper_nongithub/Figure2/RNAhybrid/resu
reslistB <- list.files(path = "D:/Krueger_Lab/Publications/miR181_paper_nongithub/Figure2/RNAhybrid/resu


#import
myfilelistA <- lapply(reslistA, Personalized_Reader)
myfilelistB <- lapply(reslistB, Personalized_Reader)




resframeA <- bind_rows(myfilelistA)
resframeB <- bind_rows(myfilelistB)



#colnames
colnames(resframeA) <- c("rownumber", "mfs", "pvalue", "start_position", "binding_bases", "non_binding_b
colnames(resframeB) <- c("rownumber", "mfs", "pvalue", "start_position", "binding_bases", "non_binding_b

resframeA[is.na(resframeA$non_binding_bases),"non_binding_bases"] <- "                        "
resframeB[is.na(resframeB$non_binding_bases),"non_binding_bases"] <- "                        "



head(resframeA)
```

```
##   rownumber   mfs   pvalue start_position
## 1         1 -13.1 1.000000             87
## 2        10 -15.7 0.999882             93
## 3       100 -19.3 0.646155             36
## 4      1000 -21.9 0.197373              4
## 5     10000 -25.4 0.026603             54
## 6     10001 -18.1 0.883059             18
##                            binding_bases                      non_binding_bases
## 1                GAGUG  G  GUC CAA                     U        CU   G   CUUACAA
## 2                    G  GCUGUC                          UGA UG      GCAACUUACAA
## 3   AGU GGCUGUCG  ACU             UACAA  UG            CA
## 4               GUGG UG    UCGCAACU   CA           UGA    C              UA   A
## 5               UGAG GGCUG CG CAAC UUACA                  U     U            A
## 6               UGAGUGGC UGU CG                                    CAACUUACAA
```

```
head(resframeB)
```

```
##   rownumber   mfs   pvalue start_position                    binding_bases
## 1         1 -10.9 1.000000             84       UUGGG   GG   GUC U
```

```
## 2        10 -23.1 0.116906             85        UUGGG  UGGC GUCGUU CUU
## 3       100 -24.3 0.059292             24      UGGGUGG UG    UCG UUACU
## 4      1000 -21.3 0.302779              5         UGGGUGG UGU CGU ACU   CA
## 5     10000 -23.8 0.078879             53        UUGGG GGCUG CGUU AC UUACA
## 6     10001 -19.2 0.711209             18  UGGGUGGC UGU CGUU       ACUU
##                   non_binding_bases
## 1            U    CU    G UACUUACAA
## 2                   U      A   ACAA
## 3     U       C              UACAA
## 4        U       C       U   UA   A
## 5             U     U            A
## 6 U                           ACAA
```

colours

```
#colours
farbeneg <- "#b4b4b4"
farbe1 <- "#0073C2FF"
farbe2 <- "#EFC000FF"
farbe3 <- "#CD534CFF"
farbe4 <- "#7AA6DCFF"
farbe5 <- "#868686FF"
farbe6 <- "#003C67FF"
farbe7 <- "#8F7700FF"
farbe8 <- "#3B3B3BFF"
farbe9 <- "#A73030FF"
farbe10 <- "#4A6990FF"
farbe11 <- "#FF6F00FF"
farbe12 <- "#C71000FF"
farbe13 <- "#008EA0FF"
farbe14 <- "#8A4198FF"
farbe15 <- "#5A9599FF"
farbe16 <- "#FF6348FF"

RNApcol <- "#b56504"
RNAncol <- "#027d73"
RPFpcol <- "#c4c404"
RPFncol <- "#8d0391"
```

Process data (remove gaps)

Due to the loops in the mRNA there are additional spaces in the mirna. We only want the binding and non binding bases of hte mirna in te correct order. For that we will remove all gaps that origin in the mRNA loops.

```
#binding and non binding bases as characters in a list
Alistbb <- strsplit(resframeA$binding_bases,"")
Alistnb <- strsplit(resframeA$non_binding_bases,"")

Blistbb <- strsplit(resframeB$binding_bases,"")
Blistnb <- strsplit(resframeB$non_binding_bases,"")
```

```r
#combine the two lists
Alist <- Map(cbind, Alistbb, Alistnb)
```

```
## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)
```

```r
Alist <- lapply(Alist, as.data.frame)

Blist <- Map(cbind, Blistbb, Blistnb)
```

```
## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)

## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)
```

```
## Warning in cbind(...): number of rows of result is not a multiple of vector
## length (arg 2)
```

```
Blist <- lapply(Blist, as.data.frame)
```

```
#remove all empty rows (mRNA loops)
Alist0 <- lapply(Alist, function(x){
  x[!(x[,1]== " " & x[,2] == " "),]
})

Blist0 <- lapply(Blist, function(x){
  x[!(x[,1]== " " & x[,2] == " "),]
})
```

```
#rewrite as characters
AlistF <- lapply(Alist0, function(x){
  paste(x[,1],  collapse = '')
})

BlistF <- lapply(Blist0, function(x){
  paste(x[,1],  collapse = '')
})
```

```
#Attach lists back onto original data.frame as new column
resframeA$binding_nospace <-unlist(AlistF)
head(resframeA$binding_nospace)
```

```
## [1] " GAGUGG  GUC CAA       " "   G  GCUGUC           "
## [3] "  AGUGGCUGUCG  ACUUACAA" "   GUGG UGUCGCAACU  CA "
## [5] "UGAG GGCUG CGCAACUUACA " "UGAGUGGCUGUCG          "
```

```
resframeB$binding_nospace <-unlist(BlistF)
head(resframeB$binding_nospace)
```

```
## [1] "UUGGG GG  GUC U         " "UUGGGUGGC GUCGUU CUU    "
## [3] " UGGGUGG UGUCGUUACU     " " UGGGUGG UGUCGU ACU  CA "
## [5] "UUGGG GGCUG CGUUACUUACA " " UGGGUGGCUGUCGUUACUU    "
```

Transform into Numbers

add 0s

replace all gaps with 0 and all letters with 1

```
#0
resframeA$binding_nospace <- chartr(" ", "0", resframeA$binding_nospace)
resframeB$binding_nospace <- chartr(" ", "0", resframeB$binding_nospace)



#1
resframeA$binding_nospace <- mgsub::mgsub(resframeA$binding_nospace, c("A", "U", "C", "G"), c(rep("1", 4
resframeB$binding_nospace <- mgsub::mgsub(resframeB$binding_nospace, c("A", "U", "C", "G"), c(rep("1", 4


head(resframeA)
```

```
##   rownumber   mfs   pvalue start_position
## 1         1 -13.1 1.000000             87
## 2        10 -15.7 0.999882             93
## 3       100 -19.3 0.646155             36
## 4      1000 -21.9 0.197373              4
## 5     10000 -25.4 0.026603             54
## 6     10001 -18.1 0.883059             18
##                            binding_bases                      non_binding_bases
## 1               GAGUG  G  GUC CAA                    U        CU   G   CUUACAA
## 2                   G  GCUGUC                        UGA UG      GCAACUUACAA
## 3   AGU GGCUGUCG  ACU             UACAA  UG            CA
## 4              GUGG UG    UCGCAACU   CA           UGA    C              UA   A
## 5              UGAG GGCUG CG CAAC UUACA                  U     U            A
## 6              UGAGUGGC UGU CG                                    CAACUUACAA
##           binding_nospace
## 1 01111110011101110000000
## 2 00010011111100000000000
## 3 00111111111110011111111
## 4 00011110111111111100110
## 5 11110111110111111111110
## 6 11111111111110000000000
```

```
head(resframeB)
```

```
##   rownumber   mfs   pvalue start_position                     binding_bases
## 1         1 -10.9 1.000000             84       UUGGG   GG   GUC U
## 2        10 -23.1 0.116906             85        UUGGG  UGGC GUCGUU CUU
## 3       100 -24.3 0.059292             24      UGGGUGG UG    UCG UUACU
## 4      1000 -21.3 0.302779              5         UGGGUGG UGU CGU ACU   CA
## 5     10000 -23.8 0.078879             53        UUGGG GGCUG CGUU AC UUACA
## 6     10001 -19.2 0.711209             18  UGGGUGGC UGU CGUU       ACUU
##                   non_binding_bases          binding_nospace
## 1            U    CU    G UACUUACAA 11111011001110100000000
## 2                   U      A   ACAA 11111111101111111011000
## 3     U       C              UACAA 01111111011111111100000
## 4        U       C       U   UA   A 01111111011111111011000
## 5             U     U             A 11111011111011111111110
```

```
## 6 U                          ACAA 01111111111111111110000
```

seperate into columns

for each base make 1 column so it can be added and also put into a heatmap

```
#for the heatmap with every binding site
heatframeA <- do.call(rbind.data.frame, strsplit(resframeA$binding_nospace,""))
heatframeA <- sapply( heatframeA, as.numeric )
colnames(heatframeA) <- c(23:1)
rownames(heatframeA) <- resframeA[,1]
head(heatframeA)
```

```
##       23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
## 1      0  1  1  1  1  1  1  0  0  1  1  1  0  1 1 1 0 0 0 0 0 0 0
## 10     0  0  0  1  0  0  1  1  1  1  1  1  0  0 0 0 0 0 0 0 0 0 0
## 100    0  0  1  1  1  1  1  1  1  1  1  1  1  0 0 1 1 1 1 1 1 1 1
## 1000   0  0  0  1  1  1  1  0  1  1  1  1  1  1 1 1 1 1 0 0 1 1 0
## 10000  1  1  1  1  0  1  1  1  1  1  0  1  1  1 1 1 1 1 1 1 1 1 0
## 10001  1  1  1  1  1  1  1  1  1  1  1  1  1  0 0 0 0 0 0 0 0 0 0
```

```
heatframeB <- do.call(rbind.data.frame, strsplit(resframeB$binding_nospace,""))
heatframeB <- sapply( heatframeB, as.numeric )
colnames(heatframeB) <- c(24:1)
rownames(heatframeB) <- resframeB[,1]
head(heatframeB)
```

```
##       24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
## 1      1  1  1  1  1  0  1  1  0  0  1  1  1  0  1 0 0 0 0 0 0 0 0 0
## 10     1  1  1  1  1  1  1  1  1  0  1  1  1  1  1 1 0 1 1 1 0 0 0 0
## 100    0  1  1  1  1  1  1  1  0  1  1  1  1  1  1 1 1 1 1 0 0 0 0 0
## 1000   0  1  1  1  1  1  1  1  0  1  1  1  1  1  1 0 1 1 1 0 0 1 1 0
## 10000  1  1  1  1  1  0  1  1  1  1  1  0  1  1  1 1 1 1 1 1 1 1 1 0
## 10001  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1 1 1 1 1 1 0 0 0 0
```

```
#reverse column order
heatframeA <-heatframeA[,23:1]
heatframeB <- heatframeB[,24:1]
```

sum of columns

```
#sum for the small heatmap with the overall binding ratio for each base
framesumA <- colSums(heatframeA)
framesumB <- colSums(heatframeB)



framesum <- as.data.frame(rbind(framesumA,framesumB))
```

```
## Warning in rbind(...): number of columns of result is not a multiple of vector
## length (arg 1)
```

```
rownames(framesum) <- c("miR181a", "miR181b")
framesum
```

```
##            1    2    3    4    5    6     7    8    9   10    11    12    13
## miR181a 1205 4404 6851 6918 7960 8676  9500 7229 5796 7656 11289 12224 11736
```

```
## miR181b 1440 4912 7353 7449 8344 9403 10549 8757 9050 9925 11709 12121 11389
##             14    15    16    17    18    19    20    21    22   23   24
## miR181a 11082 10649 10034 11382 11281 11109 11391  9738  9347 6328 1205
## miR181b 10761 10522 10129 11123 11045 11178 12424 12436 11414 8803 6027
```

```
#scale for better comperativity
sframesum <- as.data.frame(t(scale(t(framesum))))
```

Heatmap

Colours

```
hmcols1 <- c("white", "black")
hmcols2 <- colorRamp2(c(-2, 2), c("white", "red"))
```

Heatmap of all the single reads

make heatmap without column clustering but with row clustering

```
HMA <- Heatmap(heatframeA, cluster_columns = F, col = hmcols1, row_km = 5, show_row_names = F, show_row
```

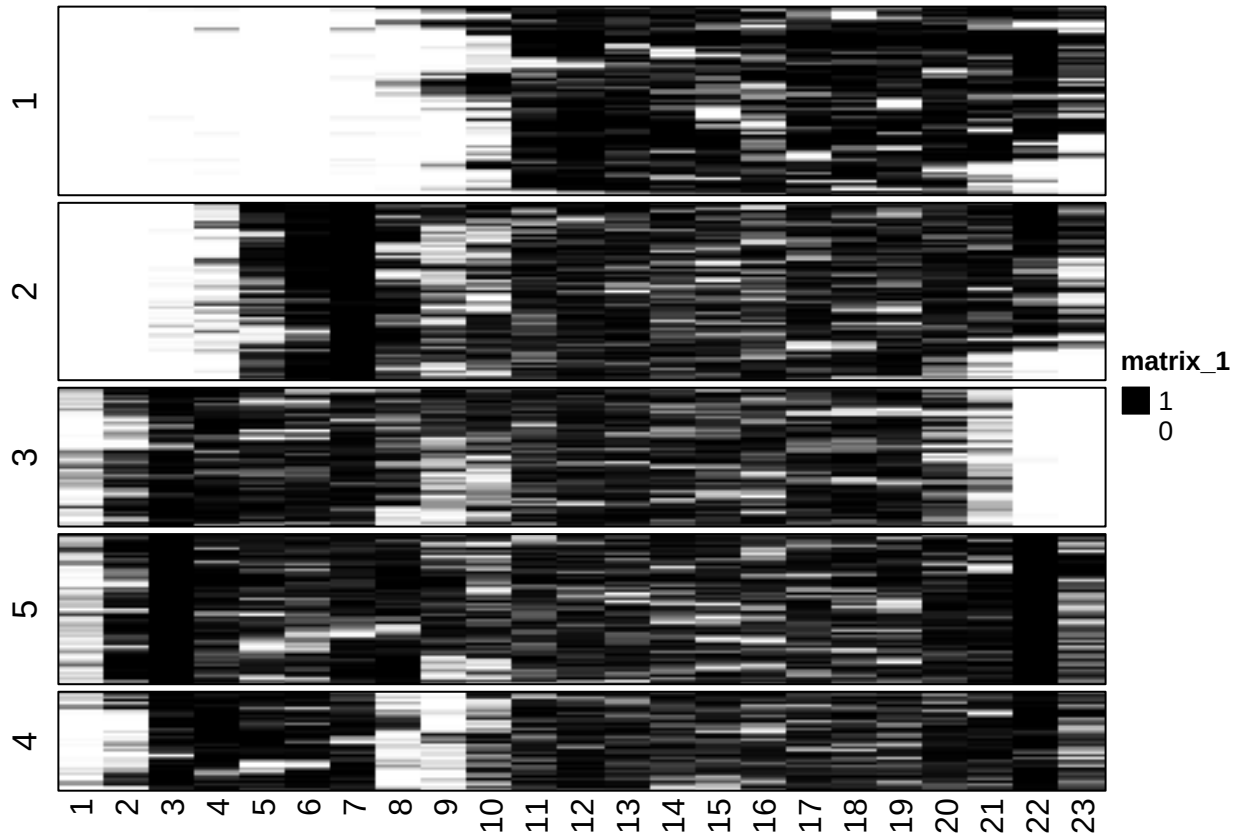
```
## `use_raster` is automatically set to TRUE for a matrix with more than
## 2000 rows. You can control `use_raster` argument by explicitly setting
## TRUE/FALSE to it.
##
## Set `ht_opt$message = FALSE` to turn off this message.
```

```
HMB <- Heatmap(heatframeB, cluster_columns = F, col = hmcols1, row_km = 5, show_row_names = F, show_row
```

```
## `use_raster` is automatically set to TRUE for a matrix with more than
## 2000 rows. You can control `use_raster` argument by explicitly setting
## TRUE/FALSE to it.
##
## Set `ht_opt$message = FALSE` to turn off this message.
```

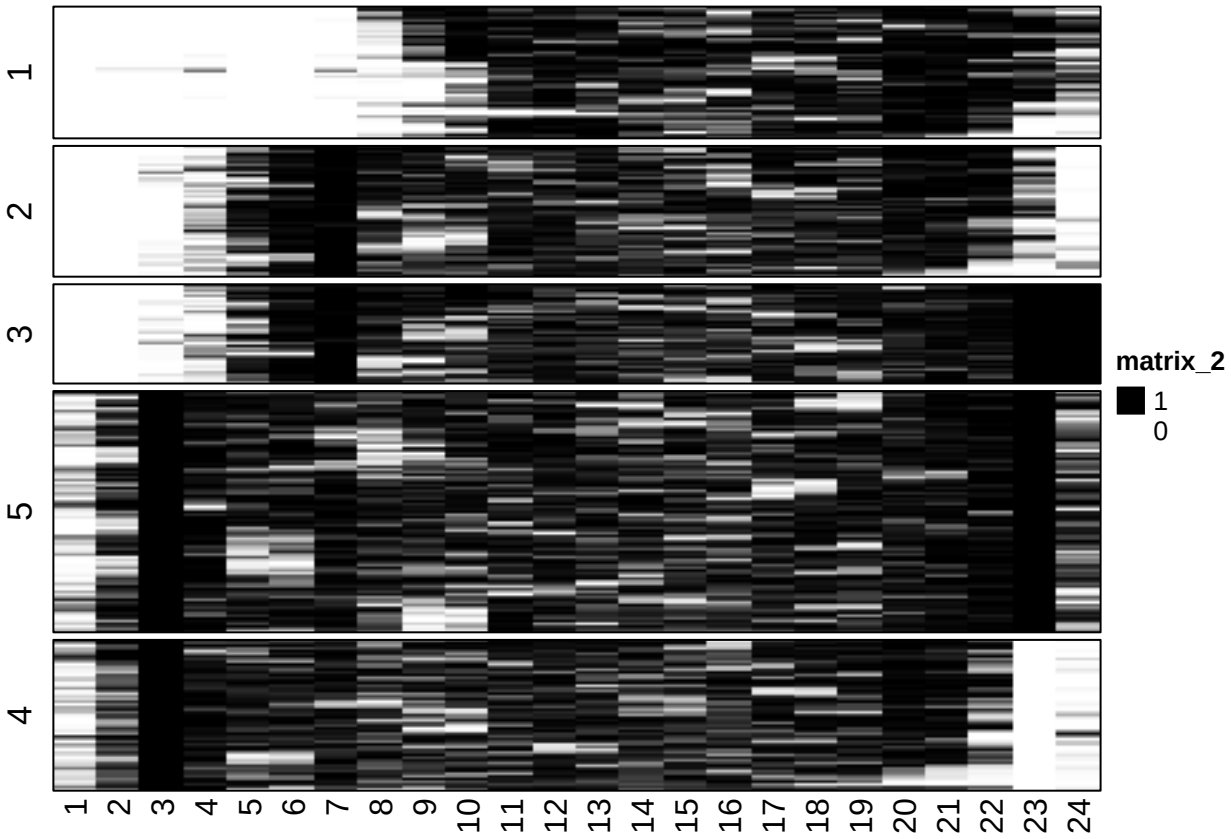
No B

```
HMA
```


1
2
3
5
4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
matrix_1
1
0

B

```
HMB
```



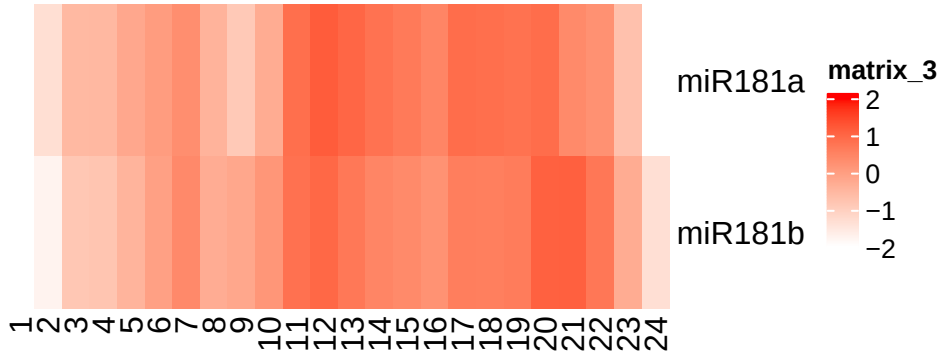
"Heatmap" of combined reads for mir_181a and b

No clustering, only sums

```
HMF <- Heatmap(sframesum, cluster_columns = F, cluster_rows = F, col = hmcols2)
```

```
## Warning: The input is a data frame-like object, convert it to a matrix.
```

```
HMF
```



cluster seperately

clustering

try to cluster seperately

```
#cluster by seed area
heat_ksA <- kmeans(heatframeA, centers = 5)
heat_k_namesA <- as.data.frame(heat_ksA$cluster)
#merge back with full data and adjust frame again
cframeA <- merge(heatframeA, heat_k_namesA, by=0)
rownames(cframeA) <- cframeA$Row.names
cframeA <- cframeA[,-1]

#order by clusters (will be needed for heatmap without clustering)
cframeA <- cframeA[order(cframeA$`heat_ksA$cluster`, decreasing = F),]
#remove cluster col
cframeA <- cframeA[,-24]
head(cframeA)
```

```
##       1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## 10000 0 1 1 1 1 1 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1
## 10003 0 0 1 1 1 1 0 0 0  1  1  1  1  1  0  0  1  1  1  1  1  1  0
## 10011 1 1 1 1 1 0 1 1 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## 10012 0 1 1 1 0 0 1 1 1  1  1  1  1  1  1  0  0  1  1  1  1  1  0
## 10016 0 0 1 1 1 1 0 0 1  1  1  0  0  1  1  1  1  1  1  1  1  1  1
## 10017 1 1 1 1 1 1 1 0 0  0  1  1  1  1  1  1  0  0  1  0  1  1  1
```

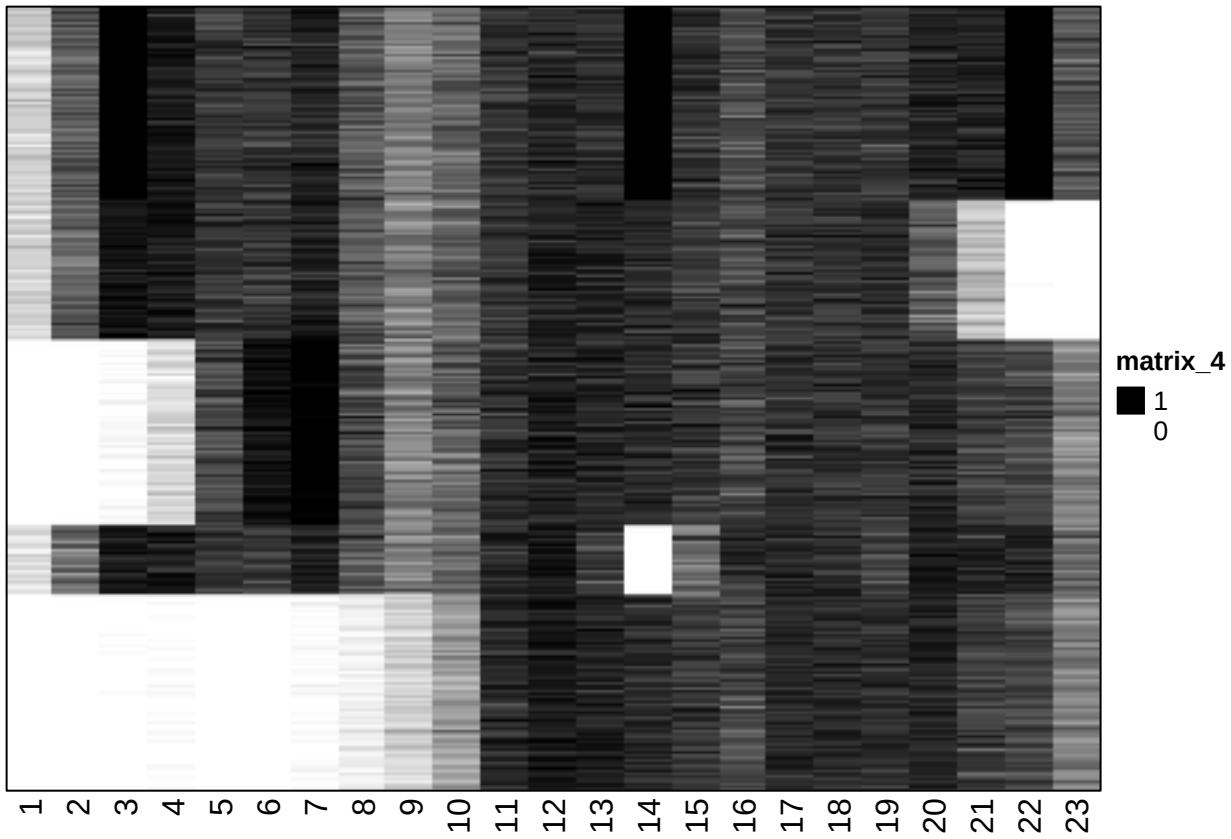
plot clustered

```
HMAsep <- Heatmap(cframeA, cluster_columns = F, cluster_rows = F, col = hmcols1, show_row_names = F, sh
```

```
## Warning: The input is a data frame-like object, convert it to a matrix.
```

```
## `use_raster` is automatically set to TRUE for a matrix with more than
## 2000 rows. You can control `use_raster` argument by explicitly setting
## TRUE/FALSE to it.
##
## Set `ht_opt$message = FALSE` to turn off this message.
```

```
HMAsep
```



clustering by seed region

try to cluster seperately only by the binding bases

```
#cluster by seed area
heat_ksAseed <- kmeans(heatframeA[,1:8], centers = 5)
heat_k_namesAseed <- as.data.frame(heat_ksAseed$cluster)
#merge back with full data and adjust frame again
cframeAseed <- merge(heatframeA, heat_k_namesAseed, by=0)
rownames(cframeAseed) <- cframeAseed$Row.names
cframeAseed <- cframeAseed[,-1]

#order by clusters (will be needed for heatmap without clustering)
cframeAseed <- cframeAseed[order(cframeAseed$`heat_ksAseed$cluster`, decreasing = F),]
#remove cluster col
cframeAseed <- cframeAseed[,-24]
head(cframeAseed)
```

```
##       1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## 100   1 1 1 1 1 1 1 1 0  0  1  1  1  1  1  1  1  1  1  1  1  0  0
## 1000  0 1 1 0 0 1 1 1 1  1  1  1  1  1  1  0  1  1  1  1  0  0  0
## 10000 0 1 1 1 1 1 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1
## 10009 0 1 1 1 1 1 1 1 1  0  1  1  1  1  1  1  1  1  1  0  0  0  0
## 10010 0 1 1 1 1 1 1 1 1  1  1  1  1  0  0  1  0  0  0  0  0  0  0
## 10015 0 1 1 1 1 1 1 1 0  0  1  1  1  1  1  1  0  0  0  0  0  0  0
```

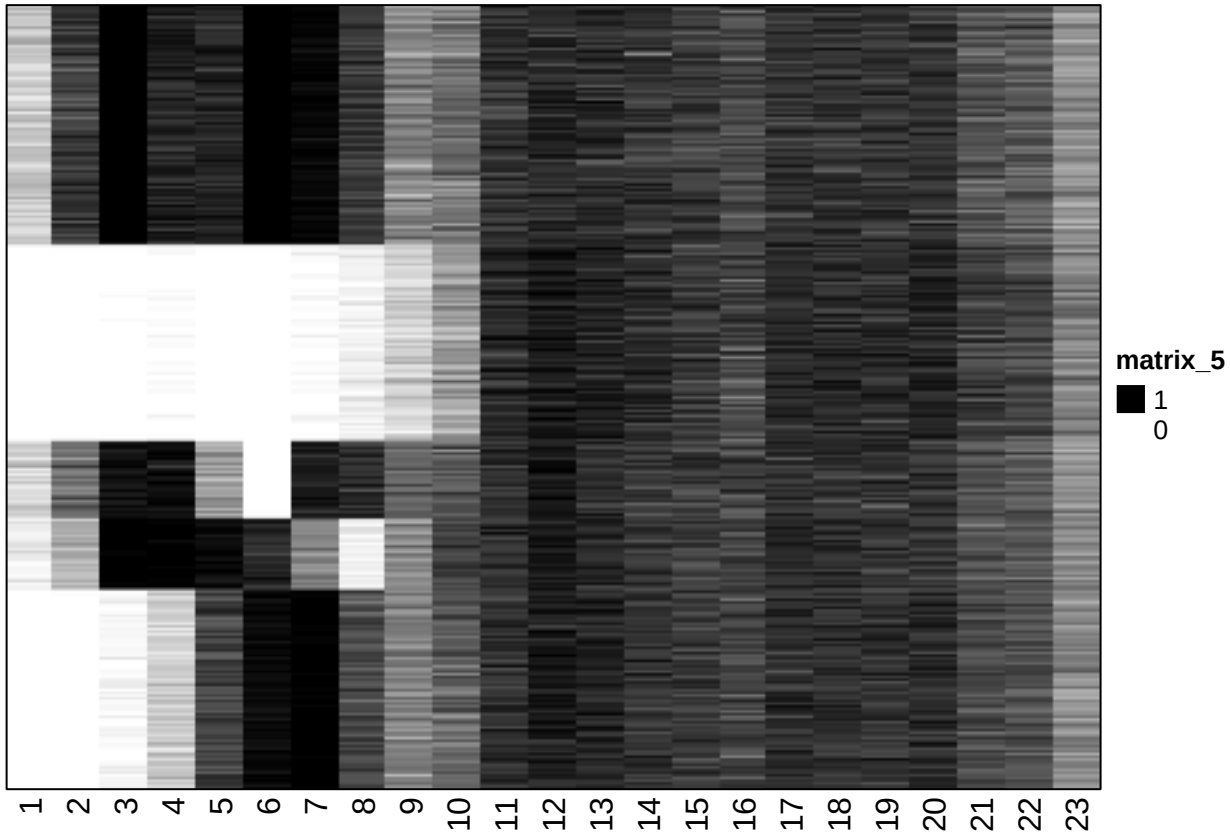
plot clustered by sed region

```
HMAseed <- Heatmap(cframeAseed, cluster_columns = F, cluster_rows = F, col = hmcols1, show_row_names = F
```

```
## Warning: The input is a data frame-like object, convert it to a matrix.
```

```
## `use_raster` is automatically set to TRUE for a matrix with more than
## 2000 rows. You can control `use_raster` argument by explicitly setting
## TRUE/FALSE to it.
##
## Set `ht_opt$message = FALSE` to turn off this message.
```

```
HMAseed
```



testcode

distframe <- dist(heatframeA) head(distframe) clustobj <- hclust(distframe)

plot(clustobj)

session info

```
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
```

```
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.utf8  LC_CTYPE=German_Germany.utf8
## [3] LC_MONETARY=German_Germany.utf8 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.utf8
##
## attached base packages:
## [1] grid      stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] ComplexHeatmap_2.15.2             circlize_0.4.15
##  [3] seqinr_4.2-30                     ggplot2_3.4.2
##  [5] dplyr_1.1.1                       BSgenome.Mmusculus.UCSC.mm10_1.4.3
##  [7] BSgenome_1.66.3                   rtracklayer_1.58.0
##  [9] Biostrings_2.66.0                 XVector_0.38.0
## [11] GenomicRanges_1.50.2              GenomeInfoDb_1.34.9
## [13] IRanges_2.32.0                    S4Vectors_0.36.2
## [15] BiocGenerics_0.44.0
##
## loaded via a namespace (and not attached):
##  [1] MatrixGenerics_1.10.0       Biobase_2.58.0
##  [3] foreach_1.5.2               highr_0.10
##  [5] GenomeInfoDbData_1.2.9      Rsamtools_2.14.0
##  [7] yaml_2.3.7                  pillar_1.9.0
##  [9] lattice_0.20-45             glue_1.6.2
## [11] digest_0.6.31               RColorBrewer_1.1-3
## [13] colorspace_2.1-0            htmltools_0.5.4
## [15] Matrix_1.5-3                XML_3.99-0.14
## [17] pkgconfig_2.0.3             GetoptLong_1.0.5
## [19] magick_2.7.4                zlibbioc_1.44.0
## [21] scales_1.2.1                BiocParallel_1.32.6
## [23] tibble_3.2.1                generics_0.1.3
## [25] withr_2.5.0                 SummarizedExperiment_1.28.0
## [27] cli_3.6.0                   magrittr_2.0.3
## [29] crayon_1.5.2                evaluate_0.20
## [31] fansi_1.0.4                 doParallel_1.0.17
## [33] MASS_7.3-58.2               Cairo_1.6-0
## [35] tools_4.2.3                 GlobalOptions_0.1.2
## [37] BiocIO_1.8.0                lifecycle_1.0.3
## [39] matrixStats_0.63.0          mgsub_1.7.3
## [41] munsell_0.5.0               cluster_2.1.4
## [43] DelayedArray_0.23.2         ade4_1.7-22
## [45] compiler_4.2.3              rlang_1.1.0
## [47] RCurl_1.98-1.12             iterators_1.0.14
## [49] rstudioapi_0.14             rjson_0.2.21
## [51] bitops_1.0-7                rmarkdown_2.21
## [53] restfulr_0.0.15             gtable_0.3.3
## [55] codetools_0.2-19            R6_2.5.1
## [57] GenomicAlignments_1.34.1    knitr_1.42
## [59] fastmap_1.1.1               utf8_1.2.3
## [61] clue_0.3-64                 shape_1.4.6
## [63] parallel_4.2.3              Rcpp_1.0.10
```

```
## [65] vctrs_0.6.1                png_0.1-8
## [67] tidyselect_1.2.0           xfun_0.37
```