ChIP-chip Using Limma Protocol

Awais Choudhry

07/09/2017

library(limma)  
library(Ringo)

## Loading required package: Biobase

## Loading required package: BiocGenerics

## Loading required package: parallel

##   
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':  
##   
## clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
## clusterExport, clusterMap, parApply, parCapply, parLapply,  
## parLapplyLB, parRapply, parSapply, parSapplyLB

## The following object is masked from 'package:limma':  
##   
## plotMA

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

## The following objects are masked from 'package:base':  
##   
## anyDuplicated, append, as.data.frame, cbind, colnames,  
## do.call, duplicated, eval, evalq, Filter, Find, get, grep,  
## grepl, intersect, is.unsorted, lapply, lengths, 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, which.max,  
## which.min

## Welcome to Bioconductor  
##   
## Vignettes contain introductory material; view with  
## 'browseVignettes()'. To cite Bioconductor, see  
## 'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: RColorBrewer

## Loading required package: Matrix

## Loading required package: grid

## Loading required package: lattice

library(edgeR)  
library(magrittr)  
library(tidyr)

##   
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':  
##   
## extract

## The following object is masked from 'package:Matrix':  
##   
## expand

library(rtracklayer)

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: S4Vectors

##   
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:tidyr':  
##   
## expand

## The following objects are masked from 'package:Matrix':  
##   
## colMeans, colSums, expand, rowMeans, rowSums

## The following objects are masked from 'package:base':  
##   
## colMeans, colSums, expand.grid, rowMeans, rowSums

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

##   
## Attaching package: 'GenomeInfoDb'

## The following objects are masked from 'package:Ringo':  
##   
## genome, genome<-

library(BSgenome.Mmusculus.UCSC.mm9)

## Loading required package: BSgenome

## Loading required package: Biostrings

## Loading required package: XVector

library(Biostrings)  
library(pander)

## Read in the Arrary Files

arrayfiles <- list.files(path="/home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent016/WholeBrain//",  
 pattern="txt.gz")  
RG <- read.maimages(arrayfiles,  
 source="agilent",  
 path="/home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent016/WholeBrain/")

## Read /home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent016/WholeBrain//GSM742103\_US45103054\_251471612167\_S01\_ChIP.txt.gz   
## Read /home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent016/WholeBrain//GSM742104\_US45103054\_251471612204\_S01\_ChIP.txt.gz   
## Read /home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent016/WholeBrain//GSM742105\_US45103054\_251471612206\_S01\_ChIP.txt.gz

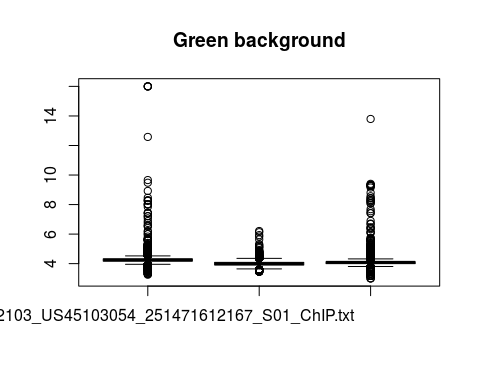
## Annotating the targets

at <- readTargets(file.path("/home/a1649239/DataFiles/ChIPseq/Mouse/","filelist.txt"))[4:6,] # the subset at the end is my edit  
RG$targets <- at  
  
RG[,1]

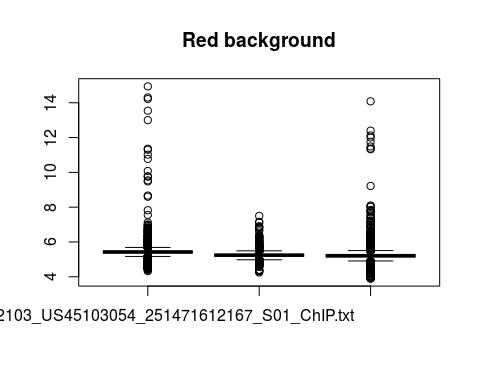
## An object of class "RGList"  
## $G  
## GSM742103\_US45103054\_251471612167\_S01\_ChIP.txt  
## [1,] 74.5  
## [2,] 36.0  
## [3,] 34.0  
## [4,] 100.0  
## [5,] 84.5  
## 243491 more rows ...  
##   
## $Gb  
## GSM742103\_US45103054\_251471612167\_S01\_ChIP.txt  
## [1,] 18  
## [2,] 19  
## [3,] 19  
## [4,] 18  
## [5,] 17  
## 243491 more rows ...  
##   
## $R  
## GSM742103\_US45103054\_251471612167\_S01\_ChIP.txt  
## [1,] 139.5  
## [2,] 49.0  
## [3,] 53.0  
## [4,] 146.0  
## [5,] 133.0  
## 243491 more rows ...  
##   
## $Rb  
## GSM742103\_US45103054\_251471612167\_S01\_ChIP.txt  
## [1,] 44  
## [2,] 44  
## [3,] 44  
## [4,] 44  
## [5,] 44  
## 243491 more rows ...  
##   
## $targets  
## Archive GSE29985\_RAW.tar  
## 4 File GSM742103\_US45103054\_251471612167\_S01\_ChIP.txt.gz  
## X01.17.2013.15.34.08 X408442880 TAR  
## 4 06/15/2011 12:19:50 26505831 TXT  
##   
## $genes  
## Row Col ControlType ProbeName SystematicName  
## 1 1 1 1 MmCGHBrightCorner MmCGHBrightCorner  
## 2 1 2 1 DarkCorner DarkCorner  
## 3 1 3 1 DarkCorner DarkCorner  
## 4 1 4 0 A\_68\_P20796006 chr1:167544862-167544919  
## 5 1 5 0 A\_68\_P25256078 chr7:078450207-078450255  
## 243491 more rows ...  
##   
## $source  
## [1] "agilent"

# BackGround Intensities

boxplot(data.frame(log2(RG$Gb)),main="Green background")



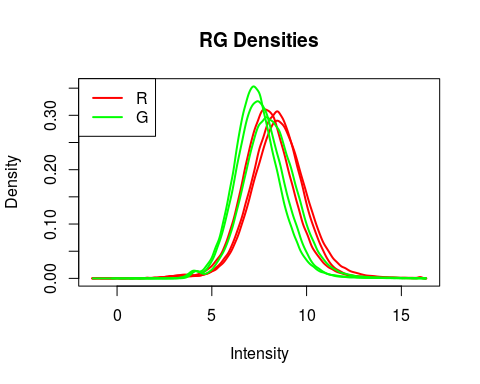
boxplot(data.frame(log2(RG$Rb)),main="Red background")



## Preprocessing

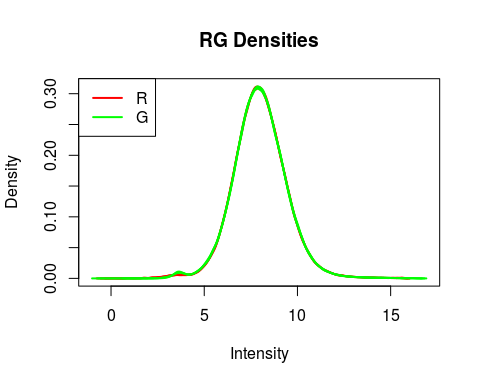
MA <- normalizeWithinArrays(RG, method="loess")  
  
plotDensities(RG)

## Warning in plotDensities.RGList(RG): NaNs produced

 Consequence of this plot we need to correct for background

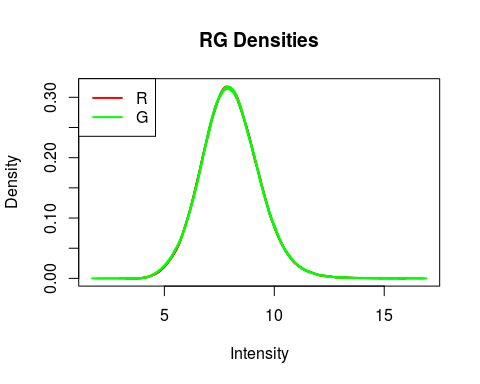
## Normalizing between Arrays

MAq <- normalizeBetweenArrays(MA, method = "Aquantile")  
 plotDensities(MAq)



## Removing Negatively/down regulated probes

MAqNegativeControlRemoved<-MAq[MAq$genes$ControlType=="0",]  
  
  
LowlyExpressedAndNegativeControlsRemoved<-MAqNegativeControlRemoved[rowSums(MAqNegativeControlRemoved$A>=4.2)>=3,]  
  
normalizedLowlyExpressedAndNegativeControlsRemoved<-normalizeBetweenArrays(LowlyExpressedAndNegativeControlsRemoved, method="Aquantile")  
  
plotDensities(normalizedLowlyExpressedAndNegativeControlsRemoved)



fit <- lmFit(normalizedLowlyExpressedAndNegativeControlsRemoved)  
fit <- eBayes(fit)  
summary(decideTests(fit))

## x1  
## -1 101  
## 0 235872  
## 1 375

ARXEnrichedRegionsWholeBrain<-topTable(fit, number = "inf")

##### ARRAY 2 Of the Whole Brain.

###################################################  
### code chunk number 26: readAgilentData  
###################################################  
arrayfiles <- list.files(path="/home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent017/WholeBrain/",  
 pattern="txt.gz")  
RG <- read.maimages(arrayfiles,  
 source="agilent",  
 path="/home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent017/WholeBrain/")

## Read /home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent017/WholeBrain//GSM742109\_US45103054\_251471712190\_S01\_ChIP.txt.gz   
## Read /home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent017/WholeBrain//GSM742110\_US45103054\_251471712233\_S01\_ChIP.txt.gz   
## Read /home/a1649239/DataFiles/ChIPseq/Mouse/ChIPAngilent017/WholeBrain//GSM742111\_US45103054\_251471712231\_S01\_ChIP.txt.gz

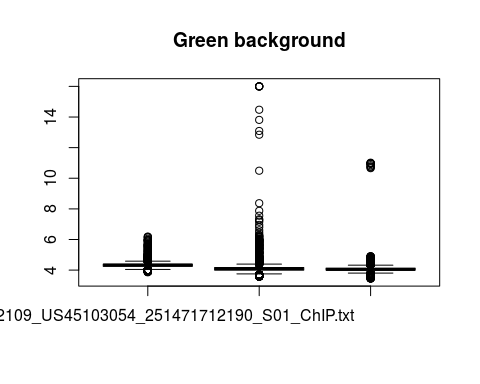
Because the ChIP-chip arrays are spilt based on different chromosomes I am calling the peaks from the same array and sample.

## Annotating the files

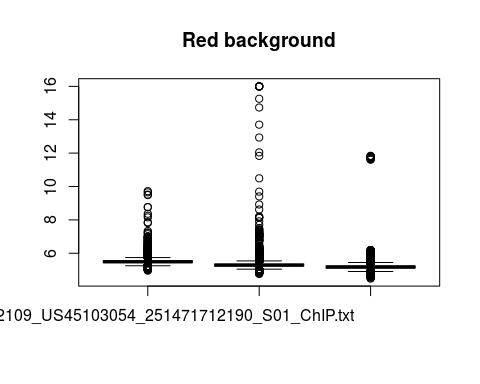
###################################################  
### code chunk number 27: readAgiTargets  
###################################################  
at <- readTargets(file.path("/home/a1649239/DataFiles/ChIPseq/Mouse/","filelist.txt"))[10:12,] # the subset at the end is my edit  
RG$targets <- at

# BackGround Intensities

boxplot(data.frame(log2(RG$Gb)),main="Green background")



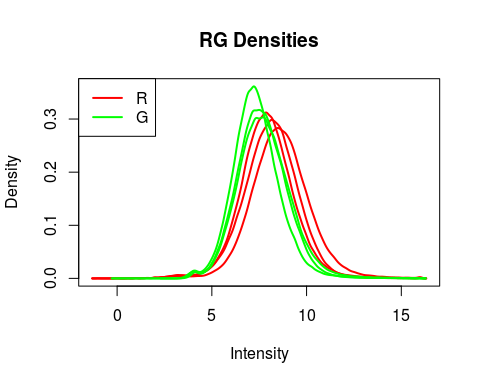
boxplot(data.frame(log2(RG$Rb)),main="Red background")



## Preprocessing

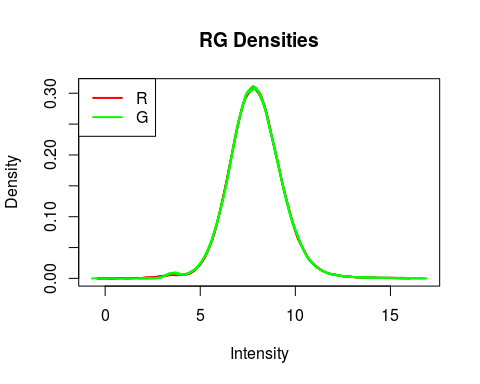
MA <- normalizeWithinArrays(RG, method="loess")  
  
plotDensities(RG)

## Warning in plotDensities.RGList(RG): NaNs produced

 Consequence of this plot we need to correct for background

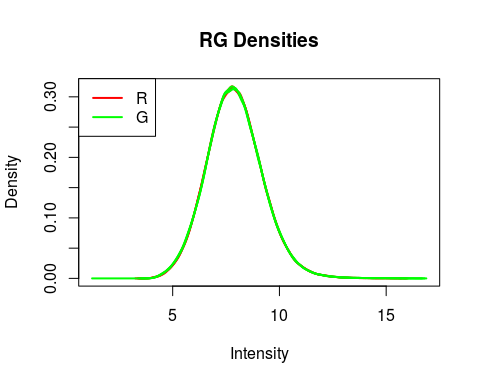
## Normalizing between Arrays

MAq <- normalizeBetweenArrays(MA, method = "Aquantile")  
 plotDensities(MAq)



## Removing Negatively/down regulated probes

MAqNegativeControlRemoved<-MAq[MAq$genes$ControlType=="0",]  
  
  
LowlyExpressedAndNegativeControlsRemoved<-MAqNegativeControlRemoved[rowSums(MAqNegativeControlRemoved$A>=4.2)>=3,]  
  
normalizedLowlyExpressedAndNegativeControlsRemoved<-normalizeBetweenArrays(LowlyExpressedAndNegativeControlsRemoved, method="Aquantile")  
  
plotDensities(normalizedLowlyExpressedAndNegativeControlsRemoved)



fit <- lmFit(normalizedLowlyExpressedAndNegativeControlsRemoved)  
fit <- eBayes(fit)  
summary(decideTests(fit))

## x1  
## -1 98  
## 0 236205  
## 1 149

ARXEnrichedRegionsWholeBrain2<-topTable(fit, number = "inf")

## Combing the Enriched Regions Controls

WholeBrainAllChIPEnrichedRanges<-rbind.data.frame(ARXEnrichedRegionsWholeBrain  
 ,ARXEnrichedRegionsWholeBrain2)  
  
significantWholeBrain<-subset(WholeBrainAllChIPEnrichedRanges, adj.P.Val<=0.001)  
significantWholeBrainEnriched<-subset(significantWholeBrain, logFC>0)  
significantWholeBrainEnriched<-subset(significantWholeBrainEnriched, SystematicName!="unmapped")  
significantWholeBrainEnriched<-separate(significantWholeBrainEnriched, col = "SystematicName",   
 into= c("chromosome", "start", "end"))

## Forming the table for homer analysis

homerTable<-cbind.data.frame("chr"=significantWholeBrainEnriched$chromosome,  
 "start"=as.numeric(significantWholeBrainEnriched$start)-200,  
 "end"=as.numeric(significantWholeBrainEnriched$end)+200,  
 "id"=1:dim(significantWholeBrainEnriched)[1],  
 "Homer",  
 "strand"= "+/-")%T>%write.table("~/DataFiles/ChIPseq/Mouse/WholeBRainHomer.bed",  
 quote=FALSE,  
 append=FALSE,  
 row.names=FALSE,  
 col.names=FALSE,  
 sep= "\t")  
  
makeGRangesFromDataFrame(homerTable, ignore.strand = TRUE)%>%reduce()

## Identifying how many of the whole Brain Peaks reanalyzed have an Arx motif in them.

## Making them 400bp+ Regions centered on the probe  
EnrichedChIPRegions<-(makeGRangesFromDataFrame(significantWholeBrainEnriched, keep.extra.columns = TRUE)+200)  
  
### Intersecting with motif sites  
JolmaSites<-matchPWM(pwm = cbind(round(PWM("TAATT")\*4), 0.25, round(PWM("ATTAA")\*4)), BSgenome.Mmusculus.UCSC.mm9, "80%")

## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them

## Warning in .Call2("PWM\_score\_starting\_at", pwm, subject, starting.at,  
## base\_codes, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them

## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them  
  
## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them

## Warning in .Call2("PWM\_score\_starting\_at", pwm, subject, starting.at,  
## base\_codes, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them

## Warning in .Call2("XString\_match\_PWM", pwm, subject, min.score,  
## count.only, : 'subject' contains letters not in [ACGT] ==> assigned weight  
## 0 to them

arx4merSites<-readRDS("~/DataFiles/ChIPseq/Mouse/ARX4mermm9Sites")  
arxSites<-readRDS("~/DataFiles/ChIPseq/Mouse/ARX6mermm9Sites")  
  
rbind.data.frame(  
"6mers In peaks"=subsetByOverlaps(EnrichedChIPRegions, arxSites)%>%length(),  
"Jolma In Peaks"= subsetByOverlaps(EnrichedChIPRegions, JolmaSites)%>%length(),  
"4mers In Peaks" = subsetByOverlaps(EnrichedChIPRegions, arx4merSites)%>%length(),  
"Total number of Peaks"= EnrichedChIPRegions%>%length()  
)%>%set\_colnames("Number Of Peaks With Motif")%>%pander()

|  |  |
| --- | --- |
|  | Number Of Peaks With Motif |
| **6mers In peaks** | 2 |
| **Jolma In Peaks** | 11 |
| **4mers In Peaks** | 27 |
| **Total number of Peaks** | 37 |