July W1

Awais Choudhry

2017-7-3

## Identifying where these clusters are commonly found

library(rtracklayer)

## Loading required package: GenomicRanges

## Loading required package: stats4

## 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 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

## Loading required package: S4Vectors

##   
## Attaching package: 'S4Vectors'

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

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

library(Biostrings)

## Loading required package: XVector

library(parallel)  
library(BSgenome.Hsapiens.UCSC.hg19)

## Loading required package: BSgenome

library(BSgenome.Mmusculus.UCSC.mm9)  
library(ggplot2)  
library(magrittr)  
library(pander)  
library(tibble)  
library(reshape2)

transponsableElements<-import("~/DataFiles/Transposable elements/Human/transponsableElement.bed")  
uniqueTransponsableElements<-as.list(unique(mcols(transponsableElements)$name))  
genome<-BSgenome.Hsapiens.UCSC.hg19  
  
  
arx6merTFBS<-matchPWM(round(PWM("TAATTA")\*7), genome, "100%")

## 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("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

arxTandem2SpacedTFBS<-matchPWM(cbind(round(PWM("TAATTA")\*7), 0.25, 0.25, round(PWM("TAATTA")\*7))  
, genome, "100%")

## 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("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

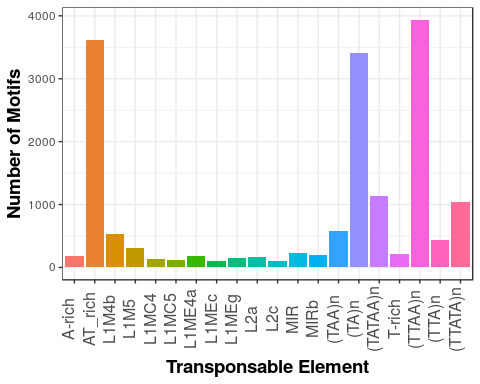
transponsableElementsFunction<-function(x){length(findOverlaps(subset(transponsableElements, name==x), arx6merTFBS))}  
  
  
mostTheConservation<-lapply(uniqueTransponsableElements, transponsableElementsFunction)  
mostTheConservation<-mostTheConservation%>%as.matrix()  
uniqueTransponsableElements<-uniqueTransponsableElements%>%as.matrix()  
colnamesTransponsable<-cbind(uniqueTransponsableElements, mostTheConservation)%>%as.data.frame()  
colnames(colnamesTransponsable)<- c("Transponsable", "Number Of Motifs")  
colnamesTransponsable$`Number Of Motifs`<-colnamesTransponsable$`Number Of Motifs`%>%as.numeric()  
colnamesTransponsable$Transponsable<-colnamesTransponsable$Transponsable%>%as.character()

## Clusters in DNA repeat elements

#Set the limit  
numberOfMotifsInArxCrm<-5  
#Find THese clusters  
arx1kb<- arx6merTFBS+500

## Warning in valid.GenomicRanges.seqinfo(x, suggest.trim = TRUE): GRanges object contains 48 out-of-bound ranges located on  
## sequences chrM, chr4\_ctg9\_hap1, chr6\_qbl\_hap6,  
## chr8\_gl000196\_random, chr8\_gl000197\_random, chr9\_gl000198\_random,  
## chr9\_gl000200\_random, chr17\_gl000203\_random,  
## chr19\_gl000209\_random, chrUn\_gl000211, chrUn\_gl000217,  
## chrUn\_gl000219, chrUn\_gl000221, chrUn\_gl000225, chrUn\_gl000232,  
## chrUn\_gl000236, chrUn\_gl000241, chrUn\_gl000242, chrUn\_gl000243,  
## chrUn\_gl000245, and chrUn\_gl000246. Note that only ranges located  
## on a non-circular sequence whose length is not NA can be  
## considered out-of-bound (use seqlengths() and isCircular() to get  
## the lengths and circularity flags of the underlying sequences).  
## You can use trim() to trim these ranges. See  
## ?`trim,GenomicRanges-method` for more information.

clustersOfARXSpecificTFBS<-subset(arx6merTFBS, countOverlaps(arx1kb, arx6merTFBS)>=numberOfMotifsInArxCrm)  
  
##Location of clusters in Repeat elements?  
transponsableElementsFunction<-function(x){length(findOverlaps(subset(transponsableElements, name==x), clustersOfARXSpecificTFBS))}  
clusterLocation<-lapply(uniqueTransponsableElements, transponsableElementsFunction)  
clusterLocation<-as.matrix(clusterLocation)  
clusteredColNames<-cbind(uniqueTransponsableElements, clusterLocation)%>%as.data.frame()  
colnames(clusteredColNames)<- c("Transponsable", "Number Of Motifs")  
clusteredColNames$`Number Of Motifs`<-clusteredColNames$`Number Of Motifs`%>%as.numeric()  
clusteredColNames$Transponsable<-clusteredColNames$Transponsable%>%as.character()  
  
ggplot(clusteredColNames[clusteredColNames$`Number Of Motifs`>=100,], aes(x=`Transponsable`, y= `Number Of Motifs`))+  
 geom\_bar(stat = "identity", aes(fill=`Transponsable`))+  
 theme\_bw()+  
 theme(axis.text.x=element\_text(size=12, vjust = -0.00,angle= 90),  
 axis.title=element\_text(size=14,face="bold"))+  
 xlab(label= "Transponsable Element")+  
 ylab(label= "Number of Motifs")+  
 guides(fill=FALSE)



clusteredColNames[clusteredColNames$`Number Of Motifs`>=10,]%>%pander

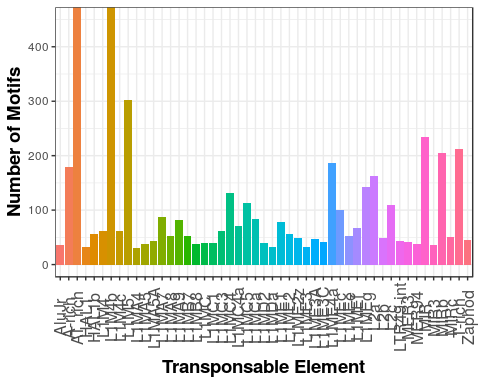
|  |  |  |
| --- | --- | --- |
|  | Transponsable | Number Of Motifs |
| **3** | L2b | 49 |
| **6** | L1MB7 | 52 |
| **8** | L2a | 163 |
| **9** | L1ME1 | 78 |
| **15** | MIRb | 205 |
| **20** | MIR3 | 36 |
| **21** | L1ME2 | 57 |
| **22** | L3 | 16 |
| **24** | L2 | 24 |
| **25** | AluJb | 26 |
| **26** | L1MEc | 101 |
| **28** | AluSx | 13 |
| **29** | MIRc | 50 |
| **32** | L4 | 27 |
| **33** | L1MA9 | 82 |
| **34** | AT\_rich | 3616 |
| **35** | L2c | 110 |
| **36** | L1MB3 | 21 |
| **39** | AluJr | 36 |
| **41** | MIR | 235 |
| **48** | AluJo | 18 |
| **53** | L1M2 | 12 |
| **55** | HAL1 | 33 |
| **58** | L1MA10 | 12 |
| **61** | L1ME3 | 32 |
| **67** | L1M4 | 62 |
| **70** | L1MDa | 33 |
| **75** | MLT1B | 15 |
| **78** | L1MC4 | 132 |
| **79** | L1ME4a | 186 |
| **81** | L1MB8 | 37 |
| **89** | (TA)n | 3411 |
| **92** | MLT1A0 | 18 |
| **94** | L1MC | 39 |
| **96** | MER5B | 14 |
| **103** | L1M3 | 12 |
| **105** | Tigger1 | 20 |
| **107** | L1MC1 | 40 |
| **110** | L1PA4 | 19 |
| **112** | MER33 | 20 |
| **113** | L1MC3 | 61 |
| **116** | L1M5 | 302 |
| **120** | L1MA1 | 11 |
| **124** | L1M4b | 532 |
| **125** | L1PB1 | 11 |
| **127** | L1MB1 | 18 |
| **130** | MLT1C | 16 |
| **133** | L1MA7 | 87 |
| **135** | L1MB2 | 29 |
| **141** | MER31-int | 22 |
| **145** | MLT1D | 13 |
| **147** | L1MA3 | 22 |
| **159** | HAL1b | 56 |
| **160** | L1MC2 | 21 |
| **165** | L1MC4a | 71 |
| **168** | L1MC5 | 113 |
| **170** | L1ME3A | 47 |
| **178** | L1MCc | 16 |
| **180** | L1MCa | 83 |
| **184** | L1ME5 | 12 |
| **186** | L1MEe | 52 |
| **188** | L1MEf | 67 |
| **190** | L1ME3C | 42 |
| **195** | L1MEd | 18 |
| **196** | A-rich | 180 |
| **198** | MER20B | 11 |
| **199** | L1MA5 | 37 |
| **200** | L1ME3D | 11 |
| **202** | L1MD | 16 |
| **206** | (CATA)n | 13 |
| **207** | L1MA8 | 52 |
| **209** | MSTD | 19 |
| **212** | L1M4c | 61 |
| **215** | L1MA4A | 10 |
| **222** | MER3 | 41 |
| **224** | MamRep434 | 22 |
| **225** | L1MA4 | 31 |
| **227** | L1ME3F | 16 |
| **229** | Charlie1a | 22 |
| **231** | L1PREC2 | 10 |
| **232** | L1MD1 | 10 |
| **234** | L1MA6 | 21 |
| **236** | LOR1-int | 10 |
| **240** | L1MD2 | 40 |
| **242** | Tigger13a | 20 |
| **250** | L1MEg | 142 |
| **252** | MER1A | 10 |
| **257** | HSAT4 | 10 |
| **258** | (TTATA)n | 1040 |
| **264** | L1MB4 | 20 |
| **272** | L1ME3B | 27 |
| **280** | MLT1A | 15 |
| **285** | Charlie5 | 22 |
| **290** | T-rich | 213 |
| **293** | (TATAA)n | 1130 |
| **302** | (CATATA)n | 44 |
| **304** | L1ME3E | 13 |
| **334** | (TATATG)n | 30 |
| **343** | MER124 | 10 |
| **345** | BLACKJACK | 24 |
| **357** | (TATG)n | 10 |
| **362** | LTR49-int | 44 |
| **376** | MER63B | 15 |
| **402** | MER4B-int | 11 |
| **430** | MER65-int | 11 |
| **433** | (TAA)n | 583 |
| **445** | L1ME2z | 49 |
| **447** | ORSL-2b | 10 |
| **472** | Charlie2b | 16 |
| **496** | HAL1-2a\_MD | 12 |
| **539** | (TTA)n | 438 |
| **544** | (TTTAA)n | 22 |
| **547** | L1MA5A | 44 |
| **624** | L1PB2 | 10 |
| **653** | (TTAA)n | 3939 |
| **663** | L1M6 | 26 |
| **668** | MER94 | 38 |
| **670** | Zaphod3 | 10 |
| **755** | Zaphod | 45 |
| **772** | Charlie2a | 15 |
| **813** | (TTAAA)n | 33 |
| **837** | Charlie13b | 14 |
| **867** | MER57A-int | 22 |
| **870** | MER101-int | 14 |
| **949** | MER61-int | 18 |
| **969** | MER4-int | 16 |
| **1037** | UCON7 | 14 |
| **1129** | (TAATG)n | 11 |
| **1158** | LTR25-int | 10 |

It is clear form thiis that the VAST majroiety of motifs are false positives and found in 'JUNK' DNA, simple repeats. Hence we will need to filter this out when proposing the tre motifs, this can be achieved either by selectively refining to enhancers and promoteres, additionally removing all motifs found in simple repeats. This will remove some true positives however, this will improve the overall accuracy.

# ARX motifs in Only transponsable elements only

At rich however, does not follow the same namiing covention hence it is retained unforunatley.

transponsableElementsOnly<-transponsableElements[!grepl( "\\(", x = transponsableElements$name)]  
  
clusterLocationOfTransponableElements<-lapply(transponsableElementsOnly$name%>%unique(),  
 transponsableElementsFunction)  
  
clusterLocationOfTransponableElements<-as.matrix(clusterLocationOfTransponableElements)  
colnamesClusterLocationOfTransponableElements<-cbind(transponsableElementsOnly$name%>%unique(), clusterLocationOfTransponableElements)%>%as.data.frame()  
colnames(colnamesClusterLocationOfTransponableElements)<- c("Transponsable", "Number Of Motifs")  
colnamesClusterLocationOfTransponableElements$`Number Of Motifs`<-colnamesClusterLocationOfTransponableElements$`Number Of Motifs`%>%as.numeric()  
colnamesClusterLocationOfTransponableElements$Transponsable<-colnamesClusterLocationOfTransponableElements$Transponsable%>%as.character()  
  
ggplot(colnamesClusterLocationOfTransponableElements[colnamesClusterLocationOfTransponableElements$`Number Of Motifs`>=30,], aes(x=`Transponsable`, y= `Number Of Motifs`))+  
 geom\_bar(stat = "identity", aes(fill=`Transponsable`))+  
 theme\_bw()+  
 theme(axis.text.x=element\_text(size=12, vjust = 0.50,angle= 90),  
 axis.title=element\_text(size=14,face="bold"))+  
 xlab(label= "Transponsable Element")+  
 ylab(label= "Number of Motifs")+  
 guides(fill=FALSE)+  
 coord\_cartesian(ylim = c(0, 450))



colnamesClusterLocationOfTransponableElements[colnamesClusterLocationOfTransponableElements$`Number Of Motifs`>=10,]%>%pander

|  |  |  |
| --- | --- | --- |
|  | Transponsable | Number Of Motifs |
| **3** | L2b | 49 |
| **6** | L1MB7 | 52 |
| **8** | L2a | 163 |
| **9** | L1ME1 | 78 |
| **15** | MIRb | 205 |
| **20** | MIR3 | 36 |
| **21** | L1ME2 | 57 |
| **22** | L3 | 16 |
| **24** | L2 | 24 |
| **25** | AluJb | 26 |
| **26** | L1MEc | 101 |
| **28** | AluSx | 13 |
| **29** | MIRc | 50 |
| **32** | L4 | 27 |
| **33** | L1MA9 | 82 |
| **34** | AT\_rich | 3616 |
| **35** | L2c | 110 |
| **36** | L1MB3 | 21 |
| **39** | AluJr | 36 |
| **41** | MIR | 235 |
| **48** | AluJo | 18 |
| **53** | L1M2 | 12 |
| **55** | HAL1 | 33 |
| **58** | L1MA10 | 12 |
| **60** | L1ME3 | 32 |
| **66** | L1M4 | 62 |
| **69** | L1MDa | 33 |
| **74** | MLT1B | 15 |
| **77** | L1MC4 | 132 |
| **78** | L1ME4a | 186 |
| **80** | L1MB8 | 37 |
| **89** | MLT1A0 | 18 |
| **91** | L1MC | 39 |
| **93** | MER5B | 14 |
| **100** | L1M3 | 12 |
| **102** | Tigger1 | 20 |
| **103** | L1MC1 | 40 |
| **106** | L1PA4 | 19 |
| **108** | MER33 | 20 |
| **109** | L1MC3 | 61 |
| **112** | L1M5 | 302 |
| **116** | L1MA1 | 11 |
| **120** | L1M4b | 532 |
| **121** | L1PB1 | 11 |
| **123** | L1MB1 | 18 |
| **126** | MLT1C | 16 |
| **129** | L1MA7 | 87 |
| **131** | L1MB2 | 29 |
| **137** | MER31-int | 22 |
| **140** | MLT1D | 13 |
| **142** | L1MA3 | 22 |
| **154** | HAL1b | 56 |
| **155** | L1MC2 | 21 |
| **160** | L1MC4a | 71 |
| **163** | L1MC5 | 113 |
| **165** | L1ME3A | 47 |
| **172** | L1MCc | 16 |
| **174** | L1MCa | 83 |
| **177** | L1ME5 | 12 |
| **179** | L1MEe | 52 |
| **181** | L1MEf | 67 |
| **183** | L1ME3C | 42 |
| **188** | L1MEd | 18 |
| **189** | A-rich | 180 |
| **191** | MER20B | 11 |
| **192** | L1MA5 | 37 |
| **193** | L1ME3D | 11 |
| **195** | L1MD | 16 |
| **199** | L1MA8 | 52 |
| **201** | MSTD | 19 |
| **204** | L1M4c | 61 |
| **207** | L1MA4A | 10 |
| **214** | MER3 | 41 |
| **216** | MamRep434 | 22 |
| **217** | L1MA4 | 31 |
| **219** | L1ME3F | 16 |
| **221** | Charlie1a | 22 |
| **223** | L1PREC2 | 10 |
| **224** | L1MD1 | 10 |
| **226** | L1MA6 | 21 |
| **228** | LOR1-int | 10 |
| **232** | L1MD2 | 40 |
| **234** | Tigger13a | 20 |
| **242** | L1MEg | 142 |
| **244** | MER1A | 10 |
| **248** | HSAT4 | 10 |
| **252** | L1MB4 | 20 |
| **259** | L1ME3B | 27 |
| **266** | MLT1A | 15 |
| **270** | Charlie5 | 22 |
| **274** | T-rich | 213 |
| **282** | L1ME3E | 13 |
| **314** | MER124 | 10 |
| **316** | BLACKJACK | 24 |
| **330** | LTR49-int | 44 |
| **339** | MER63B | 15 |
| **351** | MER4B-int | 11 |
| **367** | MER65-int | 11 |
| **377** | L1ME2z | 49 |
| **378** | ORSL-2b | 10 |
| **397** | Charlie2b | 16 |
| **417** | HAL1-2a\_MD | 12 |
| **448** | L1MA5A | 44 |
| **505** | L1PB2 | 10 |
| **541** | L1M6 | 26 |
| **544** | MER94 | 38 |
| **546** | Zaphod3 | 10 |
| **609** | Zaphod | 45 |
| **623** | Charlie2a | 15 |
| **675** | Charlie13b | 14 |
| **700** | MER57A-int | 22 |
| **703** | MER101-int | 14 |
| **771** | MER61-int | 18 |
| **789** | MER4-int | 16 |
| **841** | UCON7 | 14 |
| **935** | LTR25-int | 10 |

# Phastcon scores of clusters

##Phast Con Scores Same as above  
phastConScores<-import("~/DataFiles/Conservation/Human/hg19.100way.phastCons.bw",which =arx6merTFBS)

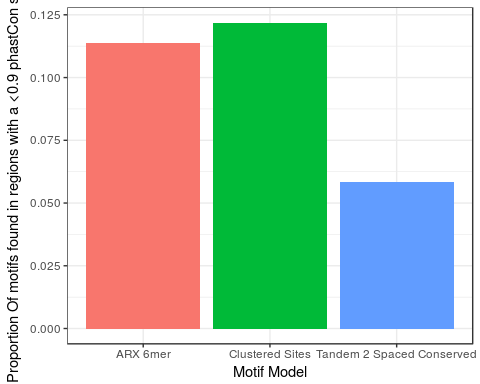
## Warning in .local(con, format, text, ...): 'which' contains seqlevels not  
## known to BigWig file: chrUn\_gl000226

PhastConTrack<-subset(phastConScores, score>=0.9)  
  
##See which Motifs fall into these regions  
phastConserved<-subset(arx6merTFBS,findOverlaps(PhastConTrack, arx6merTFBS)%>%countRnodeHits())

#PHastCon scores from previous chunnks  
clustersConserved<-subset(clustersOfARXSpecificTFBS,findOverlaps(PhastConTrack, clustersOfARXSpecificTFBS)%>%countRnodeHits())  
  
conservedArxTandem2SpacedTFBS<-subset(arxTandem2SpacedTFBS,findOverlaps(PhastConTrack, arxTandem2SpacedTFBS)%>%countRnodeHits())  
  
  
proportionOfConservedClusters<-cbind(length(conservedArxTandem2SpacedTFBS)/length(arxTandem2SpacedTFBS),   
 length(phastConserved)/ length(arx6merTFBS),  
 length(clustersConserved)/length(clustersOfARXSpecificTFBS))%>%as.data.frame()  
  
colnames(proportionOfConservedClusters)<-c("Tandem 2 Spaced Conserved", "ARX 6mer", "Clustered Sites")  
proportionOfConservedClusters<-t(proportionOfConservedClusters)%>%as.data.frame()  
proportionOfConservedClusters<- rownames\_to\_column(proportionOfConservedClusters, var= "Motif Model")%>%as.data.frame  
colnames(proportionOfConservedClusters)<-c("Motif Model", "Proportion of Motifs Conserved")  
proportionOfConservedClusters%>%pander()

|  |  |
| --- | --- |
| Motif Model | Proportion of Motifs Conserved |
| Tandem 2 Spaced Conserved | 0.05837 |
| ARX 6mer | 0.1139 |
| Clustered Sites | 0.1219 |

ggplot(proportionOfConservedClusters, aes(x= `Motif Model`, y= `Proportion of Motifs Conserved`))+   
 geom\_bar(stat="identity", aes(fill =`Motif Model`))+  
 theme\_bw()+  
 guides(fill=FALSE)+  
 ylab(label = "Proportion Of motifs found in regions with a <0.9 phastCon score")



## Classifying ARX Based on its position of Conserved motifs

We will partition the conserved motifs into four categories: adjacent (within 250 bp of the coding region of a gene), proximal (within 5 kbp of a coding region), distant (intragenic or within 100 kbp of a gene), or desert (>100 kbp from any gene).

humanTranscriponFactorStartSites<-import("~/DataFiles/Gene Tracks/Human/hg19.gtf")  
startSitesHg19<-subset(humanTranscriponFactorStartSites, type=="start\_codon")  
  
  
## Control DIstance of ALl motifs from "Codoing regions"  
distanceOfArxMotifsToTSS<-distanceToNearest(arx6merTFBS, humanTranscriponFactorStartSites)%>%as.data.frame  
## Conserved location of ARX by phastCon  
distanceOfArxMotifsToTSSConserved<-distanceToNearest(phastConserved, humanTranscriponFactorStartSites)%>%as.data.frame  
  
controlVSConservedMotifs<-cbind( rbind(  
DistantMotifs<-length(arx6merTFBS[distanceOfArxMotifsToTSS$distance>5000 & distanceOfArxMotifsToTSS$distance<=20000,])/length(arx6merTFBS),  
proxixmalMotifs<-arx6merTFBS[distanceOfArxMotifsToTSS$distance<=5000,]%>%length()/length(arx6merTFBS),  
farAwayMotifs<-arx6merTFBS[distanceOfArxMotifsToTSS$distance>20000 & distanceOfArxMotifsToTSS$distance<100000, ]%>%length()/length(arx6merTFBS),  
VeryFarAwayMotifs<-arx6merTFBS[distanceOfArxMotifsToTSS$distance>100000,]%>%length()/length(arx6merTFBS)  
), rbind(  
DistantMotifsConserved<-length(phastConserved[distanceOfArxMotifsToTSSConserved$distance>5000 & distanceOfArxMotifsToTSSConserved$distance<=20000,])/length(phastConserved),  
proxixmalMotifsConserved<-phastConserved[distanceOfArxMotifsToTSSConserved$distance<=5000,]%>%length()/length(phastConserved),  
farAwayMotifsConserved<-phastConserved[distanceOfArxMotifsToTSSConserved$distance>20000 & distanceOfArxMotifsToTSSConserved$distance<100000, ]%>%length()/length(phastConserved),  
VeryFarAwayMotifsConserved<-phastConserved[distanceOfArxMotifsToTSSConserved$distance>100000,]%>%length()/length(phastConserved)  
  
)  
)%>%as.data.frame()  
  
rownames(controlVSConservedMotifs)<-c("Distant Motifs", "Proximal Motifs", "Far away Motifs", "Very far away Motifs")  
controlVSConservedMotifs<- rownames\_to\_column(controlVSConservedMotifs)  
controlVSConservedMotifs%>%pander()

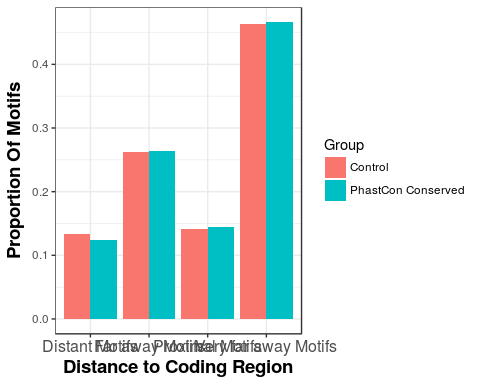
|  |  |  |
| --- | --- | --- |
| rowname | V1 | V2 |
| Distant Motifs | 0.1332 | 0.1247 |
| Proximal Motifs | 0.1412 | 0.1447 |
| Far away Motifs | 0.2624 | 0.2643 |
| Very far away Motifs | 0.4633 | 0.4663 |

# Plotting the table

# Plot stuff  
colnames(controlVSConservedMotifs)<- c("Location", "Control", "PhastCon Conserved")  
meltControlVSConservedMotifs<-melt(controlVSConservedMotifs)

## Using Location as id variables

ggplot(meltControlVSConservedMotifs, aes(x=Location, y= value))+  
 geom\_bar(stat="identity", aes(fill=variable), position = "dodge")+  
 theme\_bw()+  
 theme(axis.text.x=element\_text(size=12),  
 axis.title=element\_text(size=14,face="bold"))+  
 xlab(label= "Distance to Coding Region")+  
 ylab(label= "Proportion Of Motifs")+  
 guides(fill=guide\_legend(title="Group"))



We can see there is very little variation in the distribution of Motifs relative to protein coding regions.

# Lets see this conserved motif distribution relating to major genomic features

fantom5Enhancers<- import("~/DataFiles/Enhancer Tracks/Human/human\_permissive\_enhancers\_phase\_1\_and\_2.bed")  
hg19Genes<-import("~/DataFiles/Gene Tracks/Human/hg.bed")  
hg19Promoters<-promoters(hg19Genes)  
## Clusters in relation to genomic features  
  
ggplotDataFrameOfGenomicLocation<-rbind(  
"Enhancer"=findOverlaps(clustersOfARXSpecificTFBS, fantom5Enhancers)%>%countLnodeHits()%>%subset(clustersOfARXSpecificTFBS, .)%>%length(),  
  
"Genes"=findOverlaps(clustersOfARXSpecificTFBS, hg19Genes)%>%countLnodeHits()%>%subset(clustersOfARXSpecificTFBS, .)%>%length(),  
  
"Promoters "=findOverlaps(clustersOfARXSpecificTFBS, hg19Promoters)%>%countLnodeHits()%>%subset(clustersOfARXSpecificTFBS, .)%>%length(),  
  
"Others"=clustersOfARXSpecificTFBS%>%length()-sum( findOverlaps(clustersOfARXSpecificTFBS, fantom5Enhancers)%>%countLnodeHits()%>%subset(clustersOfARXSpecificTFBS, .)%>%length(),  
 findOverlaps(clustersOfARXSpecificTFBS, hg19Genes)%>%countLnodeHits()%>%subset(clustersOfARXSpecificTFBS, .)%>%length(),  
 findOverlaps(clustersOfARXSpecificTFBS, hg19Promoters)%>%countLnodeHits()%>%subset(clustersOfARXSpecificTFBS, .)%>%length())  
)%>%as.data.frame()%>%rownames\_to\_column()%>%as.data.frame()  
  
colnames(ggplotDataFrameOfGenomicLocation)<-c("Genomic Location", "Number Of Motifs")  
ggplotDataFrameOfGenomicLocation%>%pander()

|  |  |
| --- | --- |
| Genomic Location | Number Of Motifs |
| Enhancer | 50 |
| Genes | 13159 |
| Promoters | 538 |
| Others | 54231 |

ggplot(ggplotDataFrameOfGenomicLocation, aes(x=`Genomic Location`, y= `Number Of Motifs`))+  
 geom\_bar(stat="identity", aes(fill=`Genomic Location`))+  
 theme\_bw()+  
 theme(axis.text.x=element\_text(size=12),  
 axis.title=element\_text(size=14,face="bold"))+  
 xlab(label= "Genomic Region")+  
 ylab(label= "Number of Motifs")+  
 guides(fill=guide\_legend(title="Genomic Region"))

