Testing

Alright, let's see if the revised script makes sense.

```
source( "~/Script repository/R scripting/Matrix_Fns.R" )
```

```
## Loading required package: GenomeInfoDb
## 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, 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 object is masked from 'package:utils':
##
##     findMatches
## 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: GenomicRanges
## Loading required package: Biostrings
## Loading required package: XVector
```

```
##
## Attaching package: 'Biostrings'

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

##
## Attaching package: 'data.table'

## The following object is masked from 'package:GenomicRanges':
##
##     shift

## The following object is masked from 'package:IRanges':
##
##     shift

## The following objects are masked from 'package:S4Vectors':
##
##     first, second

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

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

##
## Attaching package: 'Biobase'
```

```
## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians

##
## Attaching package: 'GenomicAlignments'

## The following object is masked from 'package:data.table':
##
##     last

## Warning in .recacheSubclasses(def@className, def, env): undefined subclass
## "DataFrameFactor" of class "vector_OR_Vector"; definition not updated
```

```
prox <- read.delim(
  file = "C:/Users/kayle/Box/Vertinolab/McKayla Ford/Projects/R-loops_Pausing/Sequence_based_analysis/C
  header = FALSE
  )
dist <- read.delim(
  file = "C:/Users/kayle/Box/Vertinolab/McKayla Ford/Projects/R-loops_Pausing/Sequence_based_analysis/C
  header = FALSE
  )
```

```
gene_info <- rbind( prox, dist )
colnames( gene_info ) <- c( "uniq_id", "chrom", "cpg_s", "cpg_e",
                            "cpg_num", "gene_s", "gene_e",
                            "strand", "name", "ex2in", "gene_l",
                            "cpg_l", "TSS_3prime", "prime5_TSS",
                            "TSS_ex", "TSS_skew", "CGI3_skew",
                            "TES_skew", "CGI5_skew", "x2in_skew",
                            "simple_skew_class")
```

```
#my debug uses chr 19 and 22 so lets get a sample of that.
gene_info_chr <- gene_info[ ( gene_info$chrom == "chr19" | gene_info$chrom == "chr22"  ), ]
gene_info_p <- gene_info_chr[ gene_info_chr$strand == "+", ]
gene_info_m <- gene_info_chr[ gene_info_chr$strand == "-", ]

gi_p_rand <- randsamp( gene_info_p, s = 20 )
gi_m_rand <- randsamp( gene_info_m, s = 20 )

TSS_CGI_E_p <- gi_p_rand[ , c("chrom", "gene_s", "cpg_e", "uniq_id", "TSS_skew", "strand") ]

TSS_CGI_E_m <- gi_m_rand[ , c("chrom", "cpg_s", "gene_e", "uniq_id", "TSS_skew", "strand") ]

TSS_CGI_E_p
```

```
##       chrom   gene_s    cpg_e             uniq_id     TSS_skew strand
## 10176 chr19 45733438 45733699    MEIOSIN_45733438 -0.114603055      +
## 7922  chr19 44954992 44955826     CLPTM1_44954992  0.229086615      +
## 9656  chr19 12938608 12938995       CALR_12938608 -0.063716474      +
## 5990  chr19 37817410 37817886  LOC644554_37817410 -0.004532457      +
## 926   chr19 21750744 21751051     CCNYL6_21750744  0.320256819      +
## 955   chr19 54200857 54201336       RPS9_54200857  0.317303428      +
```

```
## 12540 chr19 36489626 36489831 ZNF566-AS1_36489626 -0.132602415      +
## 8606  chr19 48872420 48872671   PPP1R15A_48872420 -0.101575083      +
## 7439  chr19 40576872 40577449     SHKBP1_40576872  0.103612526      +
## 1469  chr19 56643158 56643647     SMIM17_56643158  0.275312443      +
## 7559  chr22 24555649 24556106     SNRPD3_24555649 -0.125429626      +
## 9756  chr22 43955441 43955782     SAMM50_43955441 -0.094244803      +
## 8529  chr19 36916331 36916561     ZNF568_36916331  0.014748929      +
## 10370 chr22 23145323 23145783      RAB36_23145323 -0.038559819      +
## 4228  chr22 50170730 50171198      PANX2_50170730  0.119927948      +
## 1463  chr19 58408648 58409162     ZNF584_58408648  0.275770142      +
## 4116  chr22 26429259 26429714     ASPHD2_26429259  0.126116949      +
## 9938  chr19 29942238 29942958       URI1_29942238  0.089095168      +
## 5393  chr19 10961029 10961664    SMARCA4_10961029  0.048083267      +
## 381   chr19 17933014 17933334    CCDC124_17933014  0.392550008      +
```

```
TSS_CGI_E_m
```

```
##       chrom    cpg_s   gene_e            uniq_id    TSS_skew strand
## 8402  chr19 36214531 36215084    ZNF565_36215084 -0.24190775      -
## 13264 chr19  1862257  1863579      KLF16_1863579 -0.17295977      -
## 10174 chr19  1604919  1605462     UQCR11_1605462  0.11713534      -
## 8438  chr19 39435584 39435949     RPS16_39435949  0.03622562      -
## 1111  chr19  6737228  6737580     GPR108_6737580  0.30395853      -
## 10727 chr22 23716829 23717423   GUSBP11_23717423 -0.22876514      -
## 12625 chr19 27793620 27793940 LINC00662_27793940 -0.16131446      -
## 8760  chr19 10230100 10231331     S1PR2_10231331 -0.21911683      -
## 314   chr19  5790540  5791163      DUS3L_5791163  0.40572149      -
## 12678 chr19 39539486 39540161      EID2_39540161 -0.04968391      -
## 6669  chr19 39409499 39409678   MIR4530_39409678 -0.09826329      -
## 10837 chr22 42090345 42090772    NDUFA6_42090772  0.11112038      -
## 4010  chr19 52689886 52690496     ZNF83_52690496  0.13256036      -
## 11051 chr19 19732673 19733112     ZNF14_19733112 -0.01194780      -
## 12317 chr19  5719613  5720452      LONP1_5720452 -0.07004114      -
## 5548  chr22 45975589 45977162     WNT7B_45977162  0.03706061      -
## 9986  chr19 42177152 42177306    POU2F2_42177306  0.17548723      -
## 2033  chr19 52734339 52735044    ZNF611_52735044  0.24219085      -
## 3892  chr19 19320157 19320509     SUGP1_19320509  0.13960100      -
## 11048 chr19 57947318 57947706    ZNF256_57947706  0.11685532      -
```

Now to try them out. I know my scripts are currently broken so I'll start by trialling pieces. We need to do a bam and a bigwig for each type for proseq and something non-stranded, like Zach's cut and tag.

```
Dir <- "C:/Users/kayle/Box/Vertinolab/McKayla Ford/Data/"
bam1 <- paste0( Dir, "/CutnTag_n_chip/Vertino lab data/VER00046 2021.11.17 Cut and Tag/bams/hg38_dedup/
bam2 <- paste0( Dir, "/NascentSeq/PublicData/GSE93229_Danko_2018_MCF7_PROseq/bam/bam_nodedup/MCF7_H9.ba
bw1 <-  paste0( Dir, "/CutnTag_n_chip/Vertino lab data/VER00046 2021.11.17 Cut and Tag/bigwigs/prededup
bw2p <- paste0( Dir, "/NascentSeq/PublicData/GSE93229_Danko_2018_MCF7_PROseq/bw/pauseloc/MCF7_H9_fwd.bw
bw2m <- paste0( Dir, "/NascentSeq/PublicData/GSE93229_Danko_2018_MCF7_PROseq/bw/pauseloc/MCF7_H9_rev.bw
```

Let's try bv bamvars.

```
bv1 <- bam_vars( bam1, pairedEnd = TRUE, rnorm = TRUE )
bv2 <- bam_vars( bam2, pairedEnd = FALSE, rnorm = FALSE )
```

```
bv1[[1]]
```

```
## class: BamFile
```

```
## path: C:/Users/kayle/Box/Vertinolab/McKayla.../ZS1_19_MCF7_EV_H3K27me3.dedup.bam
## index: C:/Users/kayle/Box/Vertinolab/Mc.../ZS1_19_MCF7_EV_H3K27me3.dedup.bam.bai
## isOpen: FALSE
## yieldSize: NA
## obeyQname: FALSE
## asMates: TRUE
## qnamePrefixEnd: NA
## qnameSuffixStart: NA
```

```
bv1[[2]]
```

```
## Seqinfo object with 23 sequences from an unspecified genome:
##   seqnames seqlengths isCircular genome
##   chr1      248956422       <NA>   <NA>
##   chr10     133797422       <NA>   <NA>
##   chr11     135086622       <NA>   <NA>
##   chr12     133275309       <NA>   <NA>
##   chr13     114364328       <NA>   <NA>
##   ...             ...        ...    ...
##   chr6      170805979       <NA>   <NA>
##   chr7      159345973       <NA>   <NA>
##   chr8      145138636       <NA>   <NA>
##   chr9      138394717       <NA>   <NA>
##   chrX      156040895       <NA>   <NA>
```

```
bv1[[3]]
```

```
## [1] 19203088
```

```
bv2[[1]]
```

```
## class: BamFile
## path: C:/Users/kayle/Box/Vertinolab/McKayla Ford/Data//NascentSeq.../MCF7_H9.bam
## index: C:/Users/kayle/Box/Vertinolab/McKayla Ford/Data//Nasce.../MCF7_H9.bam.bai
## isOpen: FALSE
## yieldSize: NA
## obeyQname: FALSE
## asMates: FALSE
## qnamePrefixEnd: NA
## qnameSuffixStart: NA
```

```
bv2[[2]]
```

```
## Seqinfo object with 455 sequences from an unspecified genome:
##   seqnames                seqlengths isCircular genome
##   chr1                     248956422       <NA>   <NA>
##   chr10                    133797422       <NA>   <NA>
##   chr11                    135086622       <NA>   <NA>
##   chr11_KI270721v1_random     100316       <NA>   <NA>
##   chr12                    133275309       <NA>   <NA>
##   ...                            ...        ...    ...
##   chrUn_GL000216v2            176608       <NA>   <NA>
##   chrUn_GL000218v1            161147       <NA>   <NA>
##   chrX                     156040895       <NA>   <NA>
##   chrY                      57227415       <NA>   <NA>
##   chrY_KI270740v1_random       37240       <NA>   <NA>
```

```
bv2[[3]]
```

```
## [1] NA
```

All look good.

```
chrs <- list( "chr19", "chr22" )
bed_subp <- TSS_CGI_E_p[ TSS_CGI_E_p[[1]] == "chr19", ]
bed_subm <- TSS_CGI_E_m[ TSS_CGI_E_m[[1]] == "chr19", ]
```

Let's try the dif mats.

```
a <- 150
b <- 50
bs <- 20
n <- 10

hist_center_p <- non_scaled_mat( bed_sub = bed_subp, a = a, b = b, bs = bs, method = "peak_centered" )

hist_sanch_p <- non_scaled_mat( bed_sub = bed_subp, a = a, b = b, bs = bs, method = "single_anch" )
strandvec_sanch_p <- rep( bed_subp[[6]], times = ( a + b ) / bs )

mat_list <- bi_anch_mat( bed_sub = bed_subp, n = n )
hist_scaled_p <- mat_list[[1]]
ends_matp <- mat_list[[2]]
strandvec_scaled_p <- rep( bed_subp[[6]], times = n )

hist_center_m <- non_scaled_mat( bed_sub = bed_subm, a = a, b = b, bs = bs, method = "peak_centered" )

hist_sanch_m <- non_scaled_mat( bed_sub = bed_subm, a = a, b = b, bs = bs, method = "single_anch" )
strandvec_sanch_m <- rep( bed_subm[[6]], times = ( a + b ) / bs )

mat_list <- bi_anch_mat( bed_sub = bed_subm, n = n )
hist_scaled_m <- mat_list[[1]]
ends_matm <- mat_list[[2]]
strandvec_scaled_m <- rep( bed_subm[[6]], times = n )
```

Okay, these look good, now I'm sure my reference points are correct.

```
long_hist_center_p <- melt( hist_center_p, na.rm = TRUE )
```

```
## Warning in melt(hist_center_p, na.rm = TRUE): The melt generic in data.table
## has been passed a matrix and will attempt to redirect to the relevant reshape2
## method; please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(hist_center_p). In the next version, this warning will become an
## error.
```

I see, okay. It can only handle data.tables natively and matrices are depreciated.

So... does it have to be a formal matrix or could it be a data.table?

```
long_hist_center_p <- melt(
  as.data.table( hist_center_p,
                 keep.rownames = TRUE
                 ),
  na.rm = TRUE,
```

```
  id.vars = "rn"
  )

long_hist_sanch_p <- melt(
  as.data.table( hist_sanch_p,
                 keep.rownames = TRUE
                 ),
  na.rm = TRUE,
  id.vars = "rn"
  )

long_hist_scaled_p <- melt(
  as.data.table( hist_scaled_p,
                 keep.rownames = TRUE
                 ),
  na.rm = TRUE,
  id.vars = "rn"
  )

long_ends_mat_p <- melt(
  as.data.table( ends_matp,
                 keep.rownames = TRUE
                 ),
  na.rm = TRUE,
  id.vars = "rn"
  )
###
long_hist_center_m <- melt(
  as.data.table( hist_center_m,
                 keep.rownames = TRUE
                 ),
  na.rm = TRUE,
  id.vars = "rn"
  )

long_hist_sanch_m <- melt(
  as.data.table( hist_sanch_m,
                 keep.rownames = TRUE
                 ),
  na.rm = TRUE,
  id.vars = "rn"
  )

long_hist_scaled_m <- melt(
  as.data.table( hist_scaled_m,
                 keep.rownames = TRUE
                 ),
  na.rm = TRUE,
  id.vars = "rn"
  )

long_ends_mat_m <- melt(
  as.data.table( ends_matm,
```

```
                keep.rownames = TRUE
                ),
  na.rm = TRUE,
  id.vars = "rn"
  )
```

Looks good, lets see if this causes a problem downstream.

```
long_ends_center_p <- long_hist_center_p[[3]] + bs
long_ends_sanch_p <- long_hist_sanch_p[[3]] + bs
long_ends_center_m <- long_hist_center_m[[3]] + bs
long_ends_sanch_m <- long_hist_sanch_m[[3]] + bs
```

Let's get the bws

```
bw_sub1 <- import(
      bw1, selection = GenomicSelection(
        "hg38", chrom = "chr19", colnames = "score") )
bw_sub2p <- import(
      bw2p, selection = GenomicSelection(
        "hg38", chrom = "chr19", colnames = "score") )
bw_sub2m <- import(
      bw2m, selection = GenomicSelection(
        "hg38", chrom = "chr19", colnames = "score") )
```

And bams

```
sbp1 <- ScanBamParam(
          which = GRanges(
            seqnames = "chr19",
            ranges = IRanges( start = 0, end = bv1$si@seqlengths[ bv1$si@seqnames == "chr19" ] ) ) )
sbp2 <- ScanBamParam(
          which = GRanges(
            seqnames = "chr19",
            ranges = IRanges( start = 0, end = bv2$si@seqlengths[ bv2$si@seqnames == "chr19" ] ) ) )
```

```
bam_aln1 <- GRanges( readGAlignments( bv1$bam_info, param = sbp1 ) )
bam_aln2 <- GRanges( readGAlignments( bv2$bam_info, param = sbp2 ) )
```

we need to revcomp the proseq bam. We'll skip sbp mode for now though.

```
bam_aln2 <- bam_revcomp( bam_aln2, FALSE )
```

```
gr_center_p <- GRanges(
        seqnames = "chr19",
        ranges = IRanges( start = long_hist_center_p[[3]] + 1, end = long_ends_center_p ),
        strand = strandvec_sanch_p
      )
    gr_center_p$ID <-  long_hist_center_p[[1]]
    gr_center_p$BI <-  long_hist_center_p[[2]]

gr_sanch_p <- GRanges(
        seqnames = "chr19",
        ranges = IRanges( start = long_hist_sanch_p[[3]] + 1, end = long_ends_sanch_p ),
        strand = strandvec_sanch_p
      )
    gr_sanch_p$ID <-  long_hist_sanch_p[[1]]
```

```
    gr_sanch_p$BI <-  long_hist_sanch_p[[2]]

gr_scaled_p <- GRanges(
        seqnames = "chr19",
        ranges = IRanges( start = long_hist_scaled_p[[3]] + 1, end = long_ends_mat_p[[3]] ),
        strand = strandvec_scaled_p
      )
    gr_scaled_p$ID <-  long_hist_scaled_p[[1]]
    gr_scaled_p$BI <-  long_hist_scaled_p[[2]]

gr_center_m <- GRanges(
        seqnames = "chr19",
        ranges = IRanges( start = long_hist_center_m[[3]] + 1, end = long_ends_center_m ),
        strand = strandvec_sanch_m
      )
    gr_center_m$ID <-  long_hist_center_m[[1]]
    gr_center_m$BI <-  long_hist_center_m[[2]]

gr_sanch_m <- GRanges(
        seqnames = "chr19",
        ranges = IRanges( start = long_hist_sanch_m[[3]] + 1, end = long_ends_sanch_m ),
        strand =strandvec_sanch_m
      )
    gr_sanch_m$ID <-  long_hist_sanch_m[[1]]
    gr_sanch_m$BI <-  long_hist_sanch_m[[2]]

gr_scaled_m <- GRanges(
        seqnames = "chr19",
        ranges = IRanges( start = long_hist_scaled_m[[3]] + 1, end = long_ends_mat_m[[3]] ),
        strand = strandvec_scaled_m
      )
    gr_scaled_m$ID <-  long_hist_scaled_m[[1]]
    gr_scaled_m$BI <-  long_hist_scaled_m[[2]]
```

```
tmp_mat_center_p_bam1 <- bam_olaps( gr_center_p, bam_aln1, ignorestrand=TRUE, long_hist_center_p, debug=
tmp_mat_center_m_bam1 <- bam_olaps( gr_center_m, bam_aln1, ignorestrand=TRUE, long_hist_center_m, debug=

tmp_mat_sanch_p_bam1 <- bam_olaps( gr_sanch_p, bam_aln1, ignorestrand=TRUE, long_hist_sanch_p, debug=FA
tmp_mat_sanch_m_bam1 <- bam_olaps( gr_sanch_m, bam_aln1, ignorestrand=TRUE, long_hist_sanch_m, debug=FA

tmp_mat_scaled_p_bam1 <- bam_olaps( gr_scaled_p, bam_aln1, ignorestrand=TRUE, long_hist_scaled_p, debug=
tmp_mat_scaled_m_bam1 <- bam_olaps( gr_scaled_m, bam_aln1, ignorestrand=TRUE, long_hist_scaled_m, debug=
```

Okay, do these look correct in IGV?

Why do I have one less gene in minus? Was it just too short or something? Hmm.

Center p- ZNF526_42220311 2 reads at 90bp downstream of center (note, you will have different genes as I did not set a seed for this and I random sampled.) This is at position 42,220,502. Does this hold up in IGV? Yes it does. Sanch p- 2 reads 50 bp upstream HNRNPUL1, at position 41,264,321. Yes, that also appears to match up correctly. (I probably shouldn't have used a heterochromatin marker anchored at genes to test this, though to be fair low signal is easier to count.) Scaled p- 3 reads in bin 3 of WTIP. This is located at 34,481,954 and it looks like WTIP's scaled bins are around 100 bp. Accurate. I do see I didn't calc this as pairs, but I didn't tell it to using bam aln so that's not an error in the actual script.

center m- PRKD2, +10 bp, 1 read. Position 46,716,615. Accurate. Sanch m- ZNF260, -50 bp, 2 reads. Position 36,528,301. Accurate. Scaled m- C19orf12, bin 2, 2 reads. Position 29,715,039 with 17 bp bins. Accurate.

Awesome. Now let's check non-stranded bigwig.

```
    olap <-  findOverlaps( gr_center_p, bw_sub1, ignore.strand = TRUE )
    bw_df <-  data.frame( bw_sub1[ subjectHits( olap ) ] )
    regions_df <-  data.frame( gr_center_p[ queryHits( olap ) ] )
    scored_regions <-  cbind( bw_df, regions_df )
    names( scored_regions ) <- c(
      "chrom", "bs", "be", "bin_w", "star", "score",
      names( scored_regions[ , 7:13 ] )
    )
    scored_regions$bs <- scored_regions$bs - 1 #0 index
    gs_before_bs <-  pmax( ( scored_regions$start - scored_regions$bs ), 0 )
    ge_after_be <-  pmax( ( scored_regions$be - scored_regions$end ), 0 )
    scored_regions$adjustor <- (
      scored_regions$bin_w - gs_before_bs - ge_after_be ) / scored_regions$bin_w
    scored_regions$adjusted_score <- scored_regions$score * scored_regions$adjustor
    setDT( scored_regions )
    tmp2 <- scored_regions[ ,
                            list( ( sum( adjusted_score ) / sum( adjustor ) ) ),
                            by = .( ID, BI )
                            ]
  hist <- dcast( tmp2, ID~BI, value.var = "V1" )
```

Okay good on the single now, go broad.

```
tmp_mat_center_p_bw1 <- bw_olaps( gr_center_p, bw_sub1, ignorestrand=TRUE, debug=FALSE )
tmp_mat_center_m_bw1 <- bw_olaps( gr_center_m, bw_sub1, ignorestrand=TRUE, debug=FALSE )

tmp_mat_sanch_p_bw1 <- bw_olaps( gr_sanch_p, bw_sub1, ignorestrand=TRUE, debug=FALSE )
tmp_mat_sanch_m_bw1 <- bw_olaps( gr_sanch_m, bw_sub1, ignorestrand=TRUE, debug=FALSE )

tmp_mat_scaled_p_bw1 <- bw_olaps( gr_scaled_p, bw_sub1, ignorestrand=TRUE, debug=FALSE )
tmp_mat_scaled_m_bw1 <- bw_olaps( gr_scaled_m, bw_sub1, ignorestrand=TRUE, debug=FALSE )
```

Looks like while the BW is multiplied by a value, it is in general proportional to what we got from the bam, which is good and what was expected. There's a slightly more 'smooth' transition at the edges due to overlapping bins but this effect would probably disappear in locations with more reads.

Okay, what about a stranded bam? Does that work?

```
tmp_mat_center_p_bam2 <- bam_olaps( gr_center_p, bam_aln2, ignorestrand=FALSE, long_hist_center_p, debu
tmp_mat_center_m_bam2 <- bam_olaps( gr_center_m, bam_aln2, ignorestrand=FALSE, long_hist_center_m, debu

tmp_mat_sanch_p_bam2 <- bam_olaps( gr_sanch_p, bam_aln2, ignorestrand=FALSE, long_hist_sanch_p, debug=FA
tmp_mat_sanch_m_bam2 <- bam_olaps( gr_sanch_m, bam_aln2, ignorestrand=FALSE, long_hist_sanch_m, debug=FA

tmp_mat_scaled_p_bam2 <- bam_olaps( gr_scaled_p, bam_aln2, ignorestrand=FALSE, long_hist_scaled_p, debu
tmp_mat_scaled_m_bam2 <- bam_olaps( gr_scaled_m, bam_aln2, ignorestrand=FALSE, long_hist_scaled_m, debu
```

Okay why is only center p being weird? Well I restarted R and reran this and there was no problem, so I suspect I had a cache problem or was appending to an existing value somehow. So. Centerp - 7 reads at ZNF548 -10, which is located at 57,389,991. I might call that as 8 personally, but measuring IGV by eye is hard, so algorithm is probs correct. Strand is correct, but this isn't an area with signal on both strands.

ANGPTL4 is though, and I confirmed it's correct. We'll skip centerm and sanch p because they are a bit redundant with what we've already observed and just check sanch m.Looks correct at ERF!

What about scaled, and bigwigs?

scaled - accurate values, but I do need to keep the section I had at one point that flips scaled bams bins around left to right. Check to see if that's needed for BW too before doing it.

```
tmp_mat_center_p_bw2 <- bw_olaps( gr_center_p, bw_sub2p, ignorestrand=FALSE, debug=FALSE )
tmp_mat_center_m_bw2 <- bw_olaps( gr_center_m, bw_sub2m, ignorestrand=FALSE, debug=FALSE )

tmp_mat_sanch_p_bw2 <- bw_olaps( gr_sanch_p, bw_sub2p, ignorestrand=FALSE, debug=FALSE )
tmp_mat_sanch_m_bw2 <- bw_olaps( gr_sanch_m, bw_sub2m, ignorestrand=FALSE, debug=FALSE )

tmp_mat_scaled_p_bw2 <- bw_olaps( gr_scaled_p, bw_sub2p, ignorestrand=FALSE, debug=FALSE )
tmp_mat_scaled_m_bw2 <- bw_olaps( gr_scaled_m, bw_sub2m, ignorestrand=FALSE, debug=FALSE )
```

Okay. Are they proportional to the bams?

I 'think' the center ones look a bit different but that it's only attributable to the bw being single base pair instead of fullread, not an actual formal difference. Yeah, that's definitely it. Yeah, you see the same artifact in the Bws. You also see the strand flipping problem.

I do need to flip the minus bins.