Understanding Interactions between Transcription Factors and Histone Modification in Lymphoblastoid Cell Line through ChIP-Seq Analysis

Dr. Nishat Fatima Mohammad

2025-01-02

Load Necessary packages

```
library(compGenomRData)
library(GenomeInfoDb)
library(GenomicRanges)
library(GenomicAlignments)

library(pheatmap)
library(rtracklayer)
library(Gviz)
library(ggplot2)

library(BSgenome.Hsapiens.UCSC.hg38)
library(AnnotationHub)
library(tidyr)
library(dplyr)
```

1.0 Introduction

I deeper understanding about the interplay between transcription factors like CTCF, SMC3, ZNF143, PolII (RNA polymerase 2) and histone modifications like H3K4me3, H3K36me3, H3k27ac, H3k27me3 is important to build insights into the hidden details of Lymphoblastoid cell lines and how they transform into various immune cells of diverse functions.

ChIP-Seq has empowered such analysis and this project is highly dependent on this technique to increase our understanding the above mentioned concepts.

2.0 EDA and Data Processing

Experimental data is from the public ENCODE (ENCODE Project Consortium 2012) database of ChIP-seq experiments. This Project focuses on human chromosome 21 (chr21) data only for lymphoblastoid cell line, GM12878, mapped to the GRCh38 (hg38).

```
chipseq_data_path = system.file('extdata/chip-seq',package='compGenomRData')
chipseq_files = list.files(chipseq_data_path, full.names=TRUE)
```

2.1 Clustering the Data

```
# Get chromosome lengths for hg38
hg_chromosomes = getChromInfoFromUCSC('hg38')

# Get length of chromosome 21
hg_chromosome21_len = subset(hg_chromosomes, grepl('chr21$',chrom))
hg_chromosome21_len
```

```
##    chrom     size assembled circular
## 21 chr21 46709983      TRUE    FALSE
```

Load the data from the `GenomeInfoDb` and take up Chromosome 21 from Human Genome 38. The table above shows the size among other relevant details.

2.2 Create GRanges Object

```
# Change Chromosome length data frame into vector called seqlen
seqlen = with(hg_chromosome21_len, setNames(size, chrom))

# span chr21
tillin_window = tileGenome(seqlen, tilewidth=1000)

# Get list into a GRanges object
tillin_window = unlist(tillin_window)
tillin_window
```

```
## GRanges object with 46710 ranges and 0 metadata columns:
##           seqnames            ranges strand
##              <Rle>         <IRanges>  <Rle>
##       [1]    chr21            1-1000      *
##       [2]    chr21         1001-2000      *
##       [3]    chr21         2001-3000      *
##       [4]    chr21         3001-4000      *
##       [5]    chr21         4001-5000      *
##       ...      ...               ...    ...
##   [46706]    chr21 46704985-46705984      *
##   [46707]    chr21 46705985-46706984      *
##   [46708]    chr21 46706985-46707984      *
##   [46709]    chr21 46707985-46708984      *
##   [46710]    chr21 46708985-46709983      *
##   -------
##   seqinfo: 1 sequence from an unspecified genome
```

The chromosome 21 length data frame is changed to a vector, then `tileGenome()` function from `GRanges` is used to span the chromosome with a tile width of 1000 and a Granges Object is created.

The table above shows details of the GRanges object.

2.3 Count number of Reads

```
# Get bam files
bam_file_List = list.files(path = chipseq_data_path, full.names = TRUE,
                           pattern = 'bam$')

# use summarizeOverlaps to count the reads
summary_overlap = summarizeOverlaps(tillin_window, bam_file_List)

# extract counts from SummarizedExperiment
counts = assays(summary_overlap)[[1]]
```

2.4 Normalization with Counts Per Million (CPM)

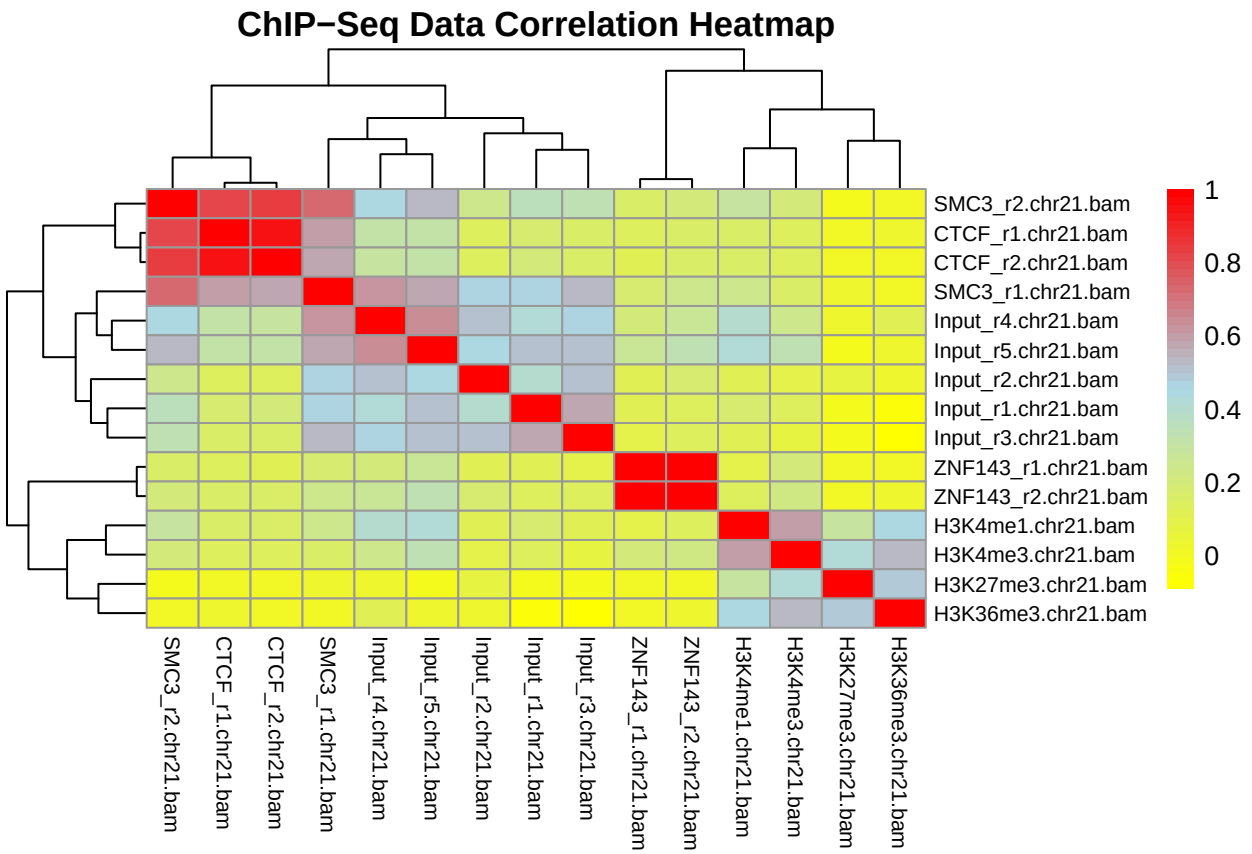
```
# Normalize with CPM
cpm = t(t(counts)*(1000000/colSums(counts)))

# Remove tiles without Overlap
cpm = cpm[rowSums(cpm) > 0,]

# remove prefix GM12878_hg38
colnames(cpm) = sub('GM12878_hg38_','',colnames(cpm))
```

Counts per Million normalizes the data by multiplying counts by 1 million and then dividing by the total number of reads.CPM is a standard way of normalizing ChIP-Seq data.

Tiles without overlap are removed and the Prefix `GM12878_hg38_` is removed.

2.5 Pearson Correlation

```
# Get pearson correlation between samples
correlation_matrix = cor(cpm, method='pearson')

# Get correlation heat map using pretty heatmap pheatmap package
pheatmap(
  mat = correlation_matrix,
  color = colorRampPalette(c("yellow", "lightblue", "red"))(50),
  cluster_rows = TRUE, cluster_cols = TRUE,
  show_rownames = TRUE, show_colnames = TRUE,
  main = "ChIP-Seq Data Correlation Heatmap",
  fontsize_row = 8,fontsize_col = 8)
```



The pairwise Pearson correlation coefficient is calculated using the `cor()` function on a region-by-sample count matrix, producing a sample-by-sample correlation matrix. This matrix is visualized as a heatmap using the `pheatmap()` function from the pretty heatmap package, which performs hierarchical clustering to group samples with high correlation. The heatmap, colored with a custom palette, highlights sample relationships and aids in quality control. For example, strong correlation between CTCF and SMC3 replicates validates expected co-localization, while the clustering of one SMC3 replicate with input samples suggests low enrichment. Biological replicates and ChIP versus input samples form distinct clusters, reflecting enrichment and experimental consistency.

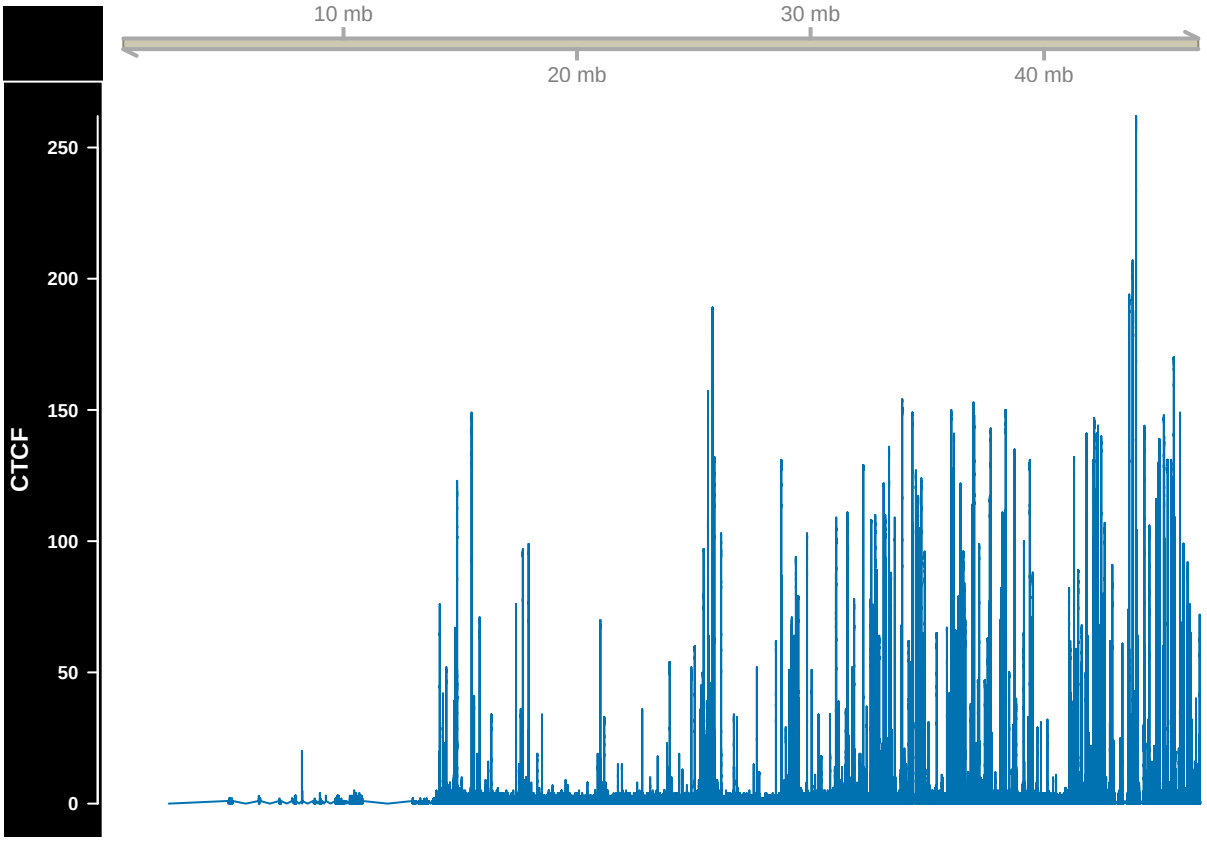
3.0 Visualization

3.1 On Browser

Visualizing ChIP-Seq data in a genome browser is a crucial step in quality assessment and preliminary analysis. Genome browsers like IGV and UCSC represent the genome as a one-dimensional coordinate system, allowing comparison of genomic annotations and experimental data in formats such as BAM, BED, and bigWig. To simplify visualization and reduce computational load, signal profiles summarizing read enrichment are generated as coverage vectors. In R, this involves importing BAM files using the `GenomicAlignments` package, converting reads to genomic ranges, extending them to an average fragment size (take 200 bp), and calculating coverage vectors with the `coverage()` function. The resulting data is exported as a bigWig file using the `rtracklayer` package. The `Gviz` package can create browser-like visualizations with customizable tracks, such as genome axis and signal profiles, plotted using the `plotTracks()` function.

```
# Get ChIp-Seq Samples
bam_file_List2 = list.files(
```

```
    path       = chipseq_data_path,
    full.names = TRUE,
    pattern    = 'bam$')

# Get the first bam file
chip_file = bam_file_List2[1]

# Get reads
reads2 = readGAlignments(chip_file)

# Get a GRanges object
reads2 = granges(reads2)

# Extend reads to 3' end
reads2 = resize(reads2, width=200, fix='start')

# Get chr21 data
reads2 = keepSeqlevels(reads2, 'chr21', pruning.mode='coarse')

# Get signal profile form read2
cov = coverage(reads2, width = seqlen)

# export the bigWig output file
export.bw(cov, (sub('.bam','.bigWig', chip_file)))


# Get genome axis track
axis= GenomeAxisTrack(
  range = GRanges('chr21', IRanges(1, width=seqlen)))

# Change signal into genomic ranges to get signal track
gcov   = as(cov, 'GRanges')
dtrack = DataTrack(gcov, name = "CTCF", type='l')

# Get track ordering
track_list = list(axis,dtrack)

# Plot tracks
plotTracks(
    trackList        = track_list,
    sizes            = c(.1,1),
    background.title = "black"
)
```



3.3 Strand Cross-correlation

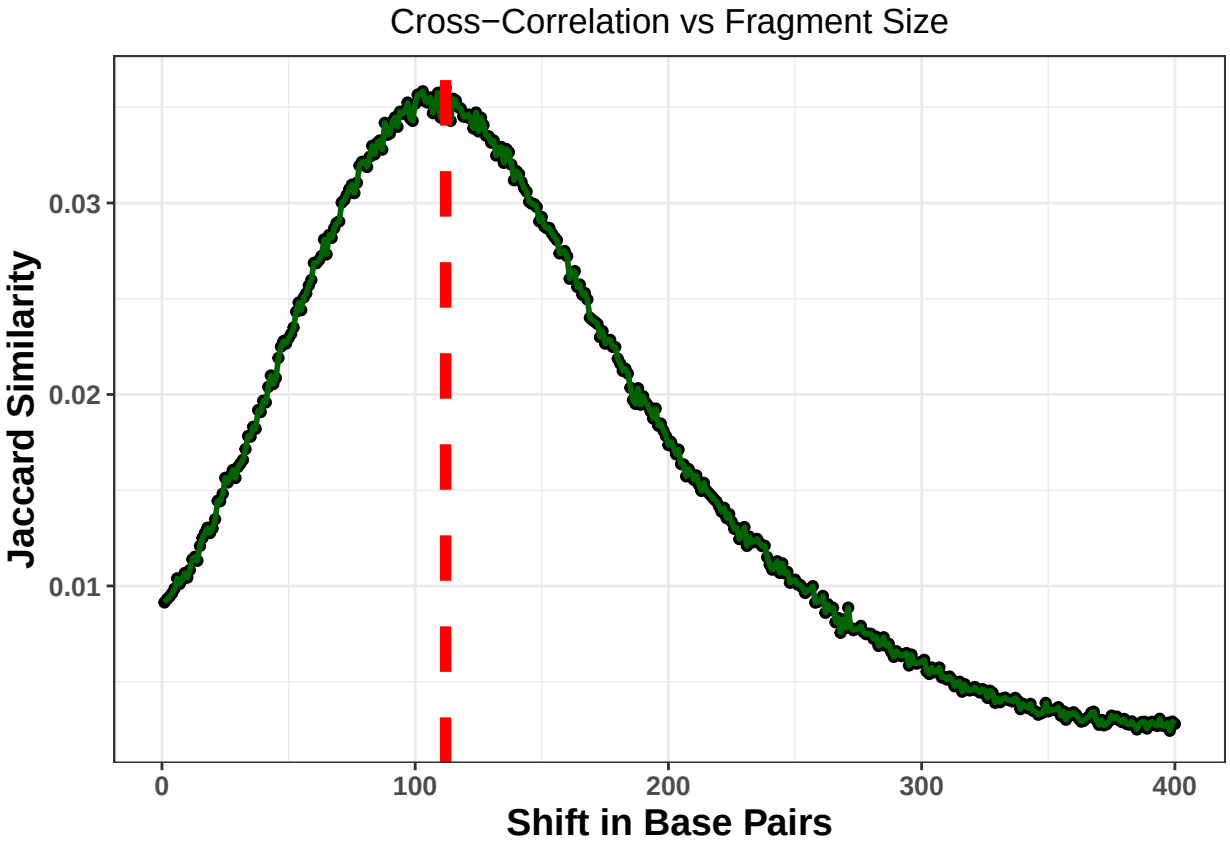
Cross-correlation between plus and minus strands quantifies whether the DNA library was enriched for fragments of specific lengths, a key indicator of ChIP-seq success. This is achieved by calculating the correlation between signal profiles of reads on the + and - strands, shifting the + strand incrementally by 1–400 base pairs. The peak of the resulting correlation curve corresponds to the average fragment length, indicating whether the ChIP enriched for fragments of a particular size.

```
# load reads
reads3 = readGAlignments(chip_file)
reads3 = granges(reads3)

# keep starting position of each read
reads3 = resize(reads3, width=1, fix='start')
reads3 = keepSeqlevels(reads3, 'chr21', pruning.mode='coarse')


# Get coverage profile for + and - strand
reads3 = split(reads3, strand(reads3))


# Get coverage vector into a boolean
cov3 = lapply(reads3, function(x){
  coverage(x, width = seqlen)[[1]] > 0})
cov3 = lapply(cov3, as.vector)
```

```r
# Get shift range
wsize = 1:400

# Get jaccard similarity
jaccard = function(x,y)sum((x & y)) / sum((x | y))

# shift + vector by 1 - 400 nucleotides and
# Get correlation coefficient
cc = shiftApply(
    SHIFT = wsize,
    X     = cov3[['+']],
    Y     = cov3[['-']],
    FUN   = jaccard)

# Get results to data frame
cc = data.frame(fragment_size = wsize, cross_correlation = cc)
head(cc)
```

```
##   fragment_size cross_correlation
## 1             1       0.009140589
## 2             2       0.009308329
## 3             3       0.009446509
## 4             4       0.009614350
## 5             5       0.009861277
## 6             6       0.010395051
```

```r
#Visualize with ggplot2
ggplot(data = cc, aes(fragment_size, cross_correlation)) +
  geom_point() +
  geom_line(color = 'darkgreen', linewidth = 1) +
  geom_vline(xintercept = which.max(cc$cross_correlation),
             size=2, color='red', linetype=2) +
  theme_bw() +
  theme(axis.text = element_text(size=10, face='bold'),
        axis.title = element_text(size=14,face="bold"),
        plot.title = element_text(hjust = 0.5)) +
  xlab('Shift in Base Pairs') +
  ylab('Jaccard Similarity') +
  ggtitle("Cross-Correlation vs Fragment Size")
```



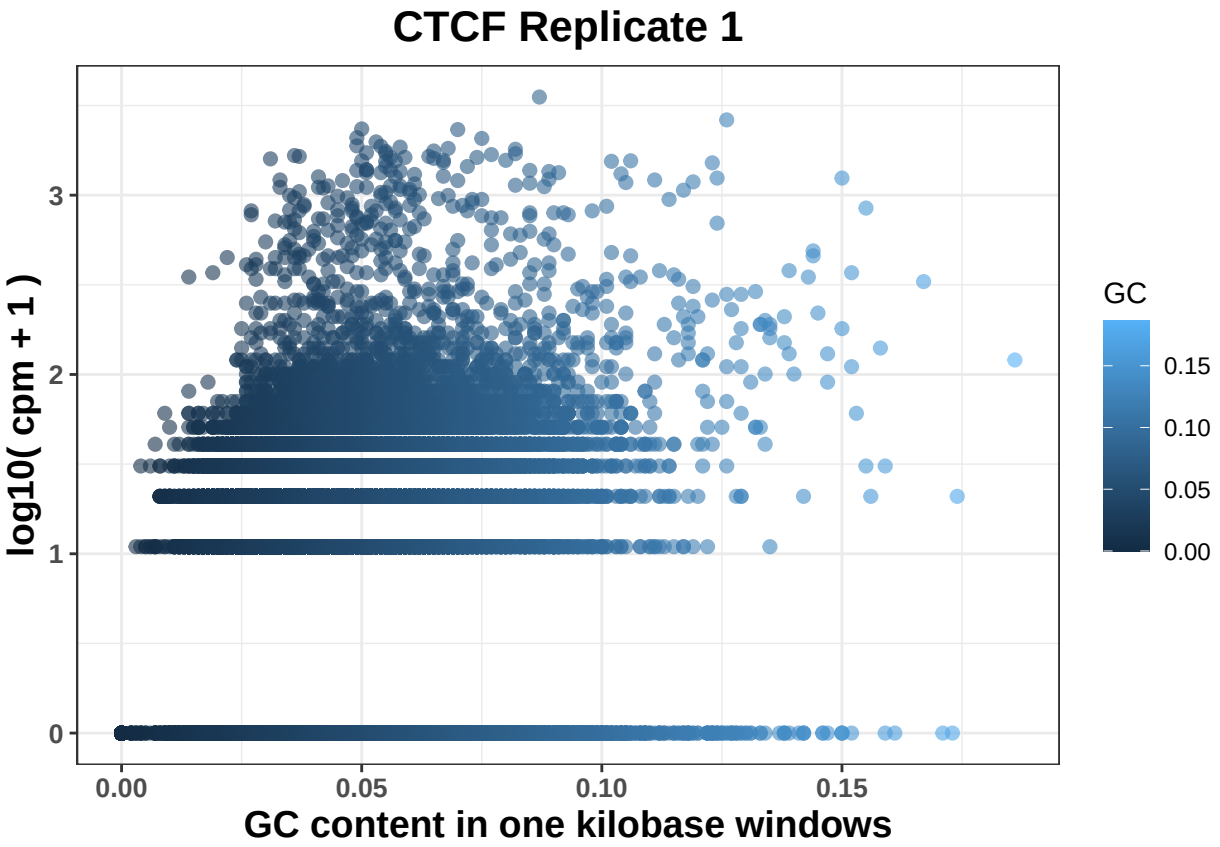
To compute this efficiently, the signal profiles are simplified into Boolean vectors representing the genomic positions of fragment ends, and the Jaccard index is used to measure similarity between shifted profiles. The resulting plot shows the correlation coefficient versus fragment size, with the peak indicating substantial enrichment of fragments, validating the ChIP-seq experiment.

4.0 Bias and Batch Effects

4.1 GC Bias Quantification

```
# Get Data
hg_chrs = getChromInfoFromUCSC('hg38')
hg_chrs = subset(hg_chrs, grepl('chr21$',chrom))
seqlength = with(hg_chrs, setNames(size, chrom))

# Get Granges Object
tilling_window = unlist(tileGenome(
    seqlengths = seqlen,
    tilewidth = 1000))

# Get sequence from genome
seq = getSeq(BSgenome.Hsapiens.UCSC.hg38, tilling_window)

# Get frequency of all dimers in sequence
nuc = oligonucleotideFrequency(seq, width = 2)
```

```r
# Change matrix to dataframe
nuc = as.data.frame(nuc)

# Get percentages
nuc = round(nuc/1000,3)

# counts reads per tilling window per experiment
so = summarizeOverlaps(tilling_window, bam_file_List2)

# CPM Normalize raw counts
counts  = assays(so)[[1]]
cpm     = t(t(counts)*(1000000/colSums(counts)))

# Transform CPM scale it using the log function
cpm_log = log10(cpm+1)

# Combine CPM values with GC Content
gc = cbind(data.frame(cpm_log), GC = nuc['GC'])

# Visualaize with ggplot2
ggplot(
    data = gc,
    aes(
        x = GC,
        y = GM12878_hg38_CTCF_r1.chr21.bam,
        color = GC
    )) +
  geom_point(size = 2, alpha = 0.6) + # Adjust alpha for better transparency
  theme_bw() +
  theme(
    axis.text  = element_text(size = 10, face = 'bold'),
    axis.title = element_text(size = 14, face = "bold"),
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5) # Adjust title font size and cente
  ) +
  xlab('GC content in one kilobase windows') +
  ylab('log10( cpm + 1 )') +
  ggtitle('CTCF Replicate 1')
```

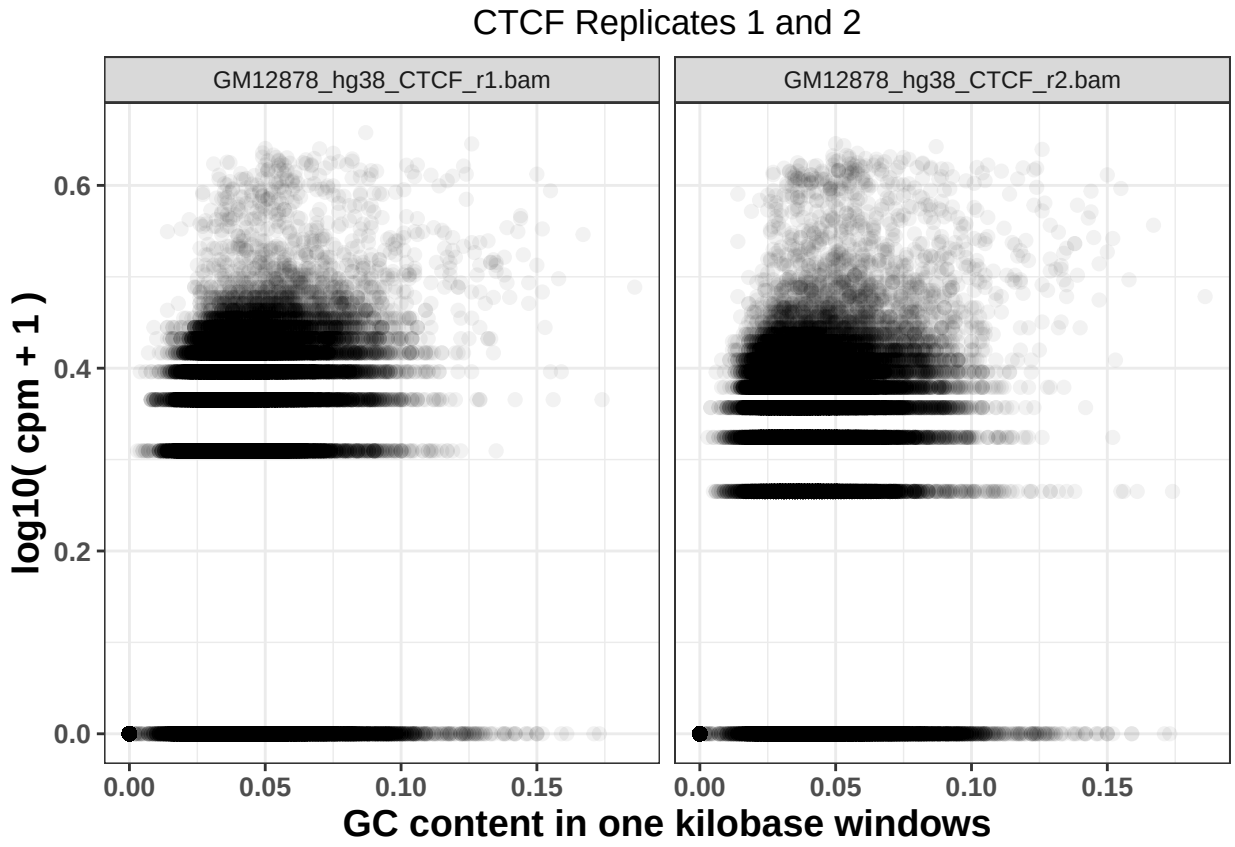


GC bias in ChIP experiments, often introduced during PCR amplification, can be evaluated by comparing GC content across tiling windows with corresponding CPM values. By calculating dinucleotide frequencies, normalizing counts, and plotting CPM versus GC content, biases in the enrichment of regions with extreme GC compositions can be identified, helping assess the success of PCR and size selection procedures.

4.2 Account for Batch Effects

```
# Reorder columns
gcd = pivot_longer(
    data      = gc,
    cols      = -GC,
    names_to  = 'experiment',
    values_to = 'cpm'
)

# Get ChIP files corresponding to ctcf experiment
gcd = subset(gcd, grepl('CTCF', experiment))

# remove chr21 suffix
gcd$experiment = sub('chr21.','',gcd$experiment)

# Visualize with ggplot2
ggplot(data = gcd, aes(GC, log10(cpm+1))) +
  geom_point(size=2, alpha=.05) +
  theme_bw() +
```

```
facet_wrap(~experiment, nrow=1)+
theme(
  axis.text = element_text(size=10, face='bold'),
  axis.title = element_text(size=14,face="bold"),
  plot.title = element_text(hjust = 0.5)) +
xlab('GC content in one kilobase windows') +
ylab('log10( cpm + 1 )') +
ggtitle('CTCF Replicates 1 and 2')
```



Comparing plots across experiments helps identify and address batch effects when ChIP-sample enrichments vary significantly between regions. By reformatting the data using pivot_longer from the tidyr package and filtering for relevant experiments, such as CTCF, the analysis isolates and visualizes the relationship between GC content and signal abundance. A scatter plot comparing GC content and CPM between CTCF replicates highlights similarities or discrepancies, enabling the identification of batch effects, which can then be accounted for in downstream analyses to ensure data consistency.

5.0 Annotation

Visualizing the genomic distribution of reads across functional regions provides insight into ChIP quality. Specific enrichment is indicated when ChIP samples differ from Input samples and align with prior knowledge of protein localization, such as H3K36me3 reads predominantly in gene bodies. To handle overlapping genomic features, a hierarchical annotation system prioritizes categories (e.g., TSS > exon > intron > intergenic) to resolve ambiguities. Genomic annotations are obtained from databases like Ensembl using tools like AnnotationHub. For example, annotations for the hg38 genome are retrieved, updated for UCSC-

style chromosome names, and subset for specific regions (e.g., chromosome 21) before defining a hierarchy for functional regions like TSS, exons, and introns.

```
# connect to the hub object
hub = AnnotationHub()

# Get human annotation from hub
AnnotationHub::query(x = hub,
                     pattern = c('ENSEMBL','Homo','GRCh38','chr','gtf'))
```

```
## AnnotationHub with 50 records
## # snapshotDate(): 2024-10-28
## # $dataprovider: Ensembl
## # $species: Homo sapiens, homo sapiens
## # $rdataclass: GRanges
## # additional mcols(): taxonomyid, genome, description,
## #   coordinate_1_based, maintainer, rdatadateadded, preparerclass, tags,
## #   rdatapath, sourceurl, sourcetype
## # retrieve records with, e.g., 'object[["AH50842"]]'
##
##              title
##   AH50842  | Homo_sapiens.GRCh38.84.chr.gtf
##   AH50843  | Homo_sapiens.GRCh38.84.chr_patch_hapl_scaff.gtf
##   AH51012  | Homo_sapiens.GRCh38.85.chr.gtf
##   AH51013  | Homo_sapiens.GRCh38.85.chr_patch_hapl_scaff.gtf
##   AH51953  | Homo_sapiens.GRCh38.86.chr.gtf
##   ...        ...
##   AH105316 | Homo_sapiens.GRCh38.107.chr_patch_hapl_scaff.gtf
##   AH110098 | Homo_sapiens.GRCh38.108.chr.gtf
##   AH110099 | Homo_sapiens.GRCh38.108.chr_patch_hapl_scaff.gtf
##   AH110867 | Homo_sapiens.GRCh38.109.chr.gtf
##   AH110868 | Homo_sapiens.GRCh38.109.chr_patch_hapl_scaff.gtf
```

```
# Get human gene annotation
gtf = hub[['AH61126']]

# Get ensemel chromosome names
ensembl_seqlevels = seqlevels(gtf)

# Add chr prefix to the chromosome names
ucsc_seqlevels = paste0('chr', ensembl_seqlevels)

# Substtute ensembl with ucsc chromosome names
seqlevels(gtf, pruning.mode='coarse') = ucsc_seqlevels

# Get chr21
gtf = gtf[seqnames(gtf) == 'chr21']

# Build a GRangesList with human annotation
annotation_list = GRangesList(tss = promoters(
  x = subset(gtf, type=='gene'),
  upstream = 1000,
  downstream = 1000),
```

```
  exon = subset(gtf, type=='exon'),
  intron = subset(gtf, type=='gene'))
```

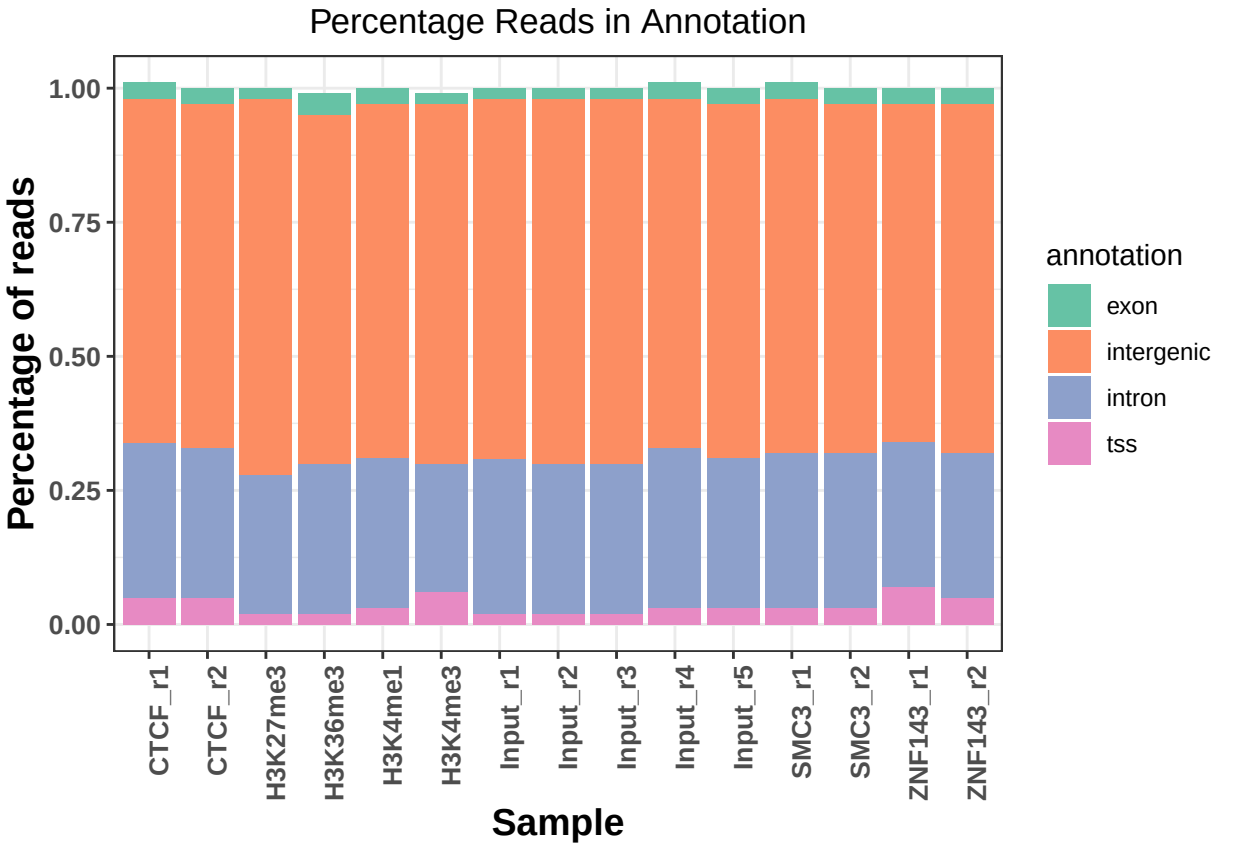
Genomic annotations can be obtained from resources like UCSC, GenBank, and Ensembl or through Bioconductor's `AnnotationHub`, which provides access to a wide range of data, including annotations for the human genome (e.g., GRCh38). Using `AnnotationHub`, users can query and download annotations in GTF format, such as the GRCh38.92 version (AH61126), stored as a `GRanges` object. After downloading, chromosome names are updated to match UCSC conventions, and data is filtered to specific chromosomes, like chromosome 21.

Annotations are then organized hierarchically into categories such as transcription start sites (TSS), exons, and introns, with intergenic regions as the lowest priority. This hierarchy is implemented using a `GRangesList` with functions like `promoters()` to define regions around the TSS and subsets for exons and introns. This structured approach ensures accurate functional categorization of genomic regions.

5.1 Annotate Reads

```
annotateReads = function(bam_file, annotation_list){

    library(dplyr)
    message(basename(bam_file))

    # load reads
    bam = readGAlignments(bam_file)

    # find overlaps between reads and annotation
    result = as.data.frame(
        findOverlaps(bam, annotation_list))

    # Add annotation index to corresponding annotation name
    annotation_name   = names(annotation_list)[result$subjectHits]
    result$annotation = annotation_name

    # Order overlaps based on hierarchy
    result = result[order(result$subjectHits),]

    # Select one category per read
    result = subset(result, !duplicated(queryHits))

    # Count reads in each category
    # group result data frame by corresponding category
    result = group_by(.data=result, annotation)

    # Count reads in each category
    result = summarise(.data = result, counts = length(annotation))

    # Classify all reads which are outside annotation as intergenic
    result = rbind(
        result,
        data.frame(annotation = 'intergenic',
                   counts= length(bam) - sum(result$counts)))
```

```
    # Get frequency
    result$frequency  = with(result, round(counts/sum(counts),2))

    # Add experiment name
    result$experiment = basename(bam_file)

    return(result)
}


# List bam files
bam_files = list.files(chipseq_data_path, full.names=TRUE, pattern='bam$')

# Get read distribution for each file
annot_reads_list = lapply(bam_files, function(x){
    annotateReads(bam_file= x, annotation_list = annotation_list)})

# collapse per file the read distributions into a dataframe
annot_reads_df = dplyr::bind_rows(annot_reads_list)

# Format experiment names
experiment_name = annot_reads_df$experiment
experiment_name = sub('.chr21.bam','', experiment_name)
experiment_name = sub('GM12878_hg38_','',experiment_name)
annot_reads_df$experiment = experiment_name


# Visualize results with ggplot2
ggplot(data = annot_reads_df,
       aes(x = experiment,
           y = frequency,
           fill = annotation)) +
    geom_bar(stat='identity') +
    theme_bw() +
    scale_fill_brewer(palette='Set2') +
    theme(axis.text = element_text(size=10, face='bold'),
          axis.title = element_text(size=14,face="bold"),
          plot.title = element_text(hjust = 0.5),
          axis.text.x = element_text(angle = 90, hjust = 1)) +
    xlab('Sample') +
    ylab('Percentage of reads') +
    ggtitle('Percentage Reads in Annotation')
```



Annotating reads involves categorizing reads from .bam files into genomic regions based on overlaps with predefined annotations. The `annotateReads()` function processes each file by loading reads, identifying overlaps, assigning categories hierarchically, counting reads per category, and calculating their frequencies, with unannotated reads classified as "intergenic." Results from all files are combined into a single data frame, reformatted, and visualized as bar plots.

The analysis highlights notable patterns, such as increased H3K36me3 reads in exons and introns and H3K4me3 reads around transcription start sites (TSS). ZNF143 transcription factor replicates also show enriched read abundance near TSS, offering insights into functional genomic distributions.

Conclusion

This concludes the QC steps and investigated the interactions between transcription factors (CTCF, SMC3, ZNF143, PolII) and histone modifications (H3K4me3, H3K36me3, H3K27ac, H3K27me3) in GM12878 cells using ChIP-Seq data. Key findings reveal strong co-localization between CTCF and SMC3, suggesting they function together in chromatin loops, while PolII and ZNF143 show variable correlations, indicating distinct transcriptional roles. Active histone marks like H3K4me3 and H3K27ac are enriched at promoters and enhancers, whereas H3K27me3 is linked to repressed regions. The analysis also confirmed dataset quality through strand cross-correlation and GC bias checks, with genome browser views illustrating the spatial distribution of these marks on chromosome 21. These insights improve our understanding of transcriptional regulation and chromatin structure in GM12878 cells. The next step will be Peak calling.

References

ENCODE Project Consortium. (2012). "An integrated encyclopedia of DNA elements in the human genome." Nature, 489(7414), 57-74. Retrieved January 02, 2025 from https://www.nature.com/articles/nature11247

Bioconductor. (2025). BSgenome.Hsapiens.UCSC.hg38: Full genomic sequences for Homo sapiens (UCSC genome hg38). Bioconductor. https://www.bioconductor.org/packages/release/data/annotation/html/BSgenome.Hsapiens.UCSC.hg38.html

Akalin, A., et al. (2020, November 26). ChIP quality control. In Computational Genomics with R. Retrieved January 02, 2025, from https://compgenomr.github.io/book/chip-quality-control.html