```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(ggplot2)
library(magick)
```

```
## Linking to ImageMagick 6.9.13.17
## Enabled features: cairo, fontconfig, freetype, ghostscript, lcms, pango, raw, rsvg, webp, x11
## Disabled features: fftw, heic
```

```r
library(stringr)
library(rtracklayer)
```

```
## Loading required package: GenomicRanges
## Loading required package: stats4
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
##
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
##
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
##
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
##     table, tapply, union, unique, unsplit, which.max, which.min
##
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
##
## The following objects are masked from 'package:lubridate':
```

```
##
##     second, second<-
##
## The following objects are masked from 'package:dplyr':
##
##     first, rename
##
## The following object is masked from 'package:tidyr':
##
##     expand
##
## 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:lubridate':
##
##     %within%
##
## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice
##
## The following object is masked from 'package:purrr':
##
##     reduce
##
## Loading required package: GenomeInfoDb
```

```
library(knitr)
library(eulerr)
library(enrichR)
```

```
## Welcome to enrichR
## Checking connections ...
## Enrichr ... Connection is Live!
## FlyEnrichr ... Connection is Live!
## WormEnrichr ... Connection is Live!
## YeastEnrichr ... Connection is Live!
## FishEnrichr ... Connection is Live!
## OxEnrichr ... Connection is Live!
```

Introduction:

RUNX1 is a transcription factor with context-dependent roles in breast cancer, where it can act as either a tumor suppressor or an oncogene, and it functions through interactions with chromatin regulators, estrogen

receptor , Polycomb complexes, and the nuclear matrix. Because breast cancer involves major alterations in nuclear structure and because RUNX1 has been linked to both gene regulation and chromatin interactions, the study was performed to determine how loss of RUNX1 affects both transcriptional programs and higher-order genome organization in MCF-7 breast cancer cells. To address these questions, the authors used RNA-seq to measure RUNX1-dependent changes in gene expression, ChIP-seq to map RUNX1 binding sites and identify direct regulatory relationships, and Hi-C to assess whether RUNX1 influences 3D chromatin architecture, including TAD boundaries and local long-range interactions. Together, these methods allowed them to connect RUNX1 binding with alterations in gene expression and genome folding.

Methodology:

ChIP-sequencing data were processed using a fully reproducible workflow implemented in Nextflow v25.04.6.5954. All tools were run with default parameters unless otherwise specified. Initial quality assessment of raw single-end FASTQ files was performed using FastQC v0.12.1 to evaluate per-base sequence quality, GC content, and adapter contamination. Adapter trimming and quality filtering were conducted using Trimmomatic v0.39, removing Illumina adapter sequences and applying quality-based trimming. A MultiQC v1.25 report was generated to aggregate FastQC metrics, Trimmomatic trimming statistics, and alignment summaries.

Reference genome indexing and read alignment were performed using Bowtie2 v2.5.4. The genome index was built from the reference genome FASTA file, and trimmed reads were aligned to produce SAM files. Aligned reads were converted to BAM format, sorted by genomic coordinates, and indexed using SAMtools v1.21. Alignment statistics were generated using samtools flagstat to assess mapping rates and read quality metrics. Signal tracks for visualization were created using deepTools v3.5.5 bamCoverage, which generated normalized bigWig files from sorted BAM files. Sample correlation analysis was performed using deepTools multiBigwigSummary to compute genome-wide signal across all IP samples, followed by plotCorrelation to generate Spearman correlation heatmaps.

Peak calling was conducted using HOMER v4.11 findPeaks in factor mode, comparing each IP sample against its corresponding input control. Tag directories were first created using makeTagDirectory from aligned BAM files. Peak positions in HOMER format were converted to BED format using pos2bed.pl and sorted using BEDtools v2.31.1. Reproducible peaks were identified by intersecting replicate peak calls and requiring peaks to be present in all replicates using bedtools intersect. Peaks overlapping ENCODE blacklist regions were removed using bedtools to exclude artifact-prone genomic regions.

Peak annotation was performed using HOMER annotatePeaks.pl, which assigned peaks to genomic features (promoter, exon, intron, intergenic) based on the provided GTF annotation and calculated distances to the nearest transcription start site. Signal enrichment profiles around gene bodies were visualized using deepTools computeMatrix in scale-regions mode, adding 2000bp of padding to both the start and end site, and with UCSC gene annotations, followed by plotProfile to generate aggregate enrichment plots across all IP samples.

De novo motif discovery was conducted using HOMER findMotifsGenome.pl on the final filtered peak set, searching for enriched sequence motifs within $\pm$200 bp of peak summits against size-matched background regions from the reference genome.

Quality Control Evaluation:

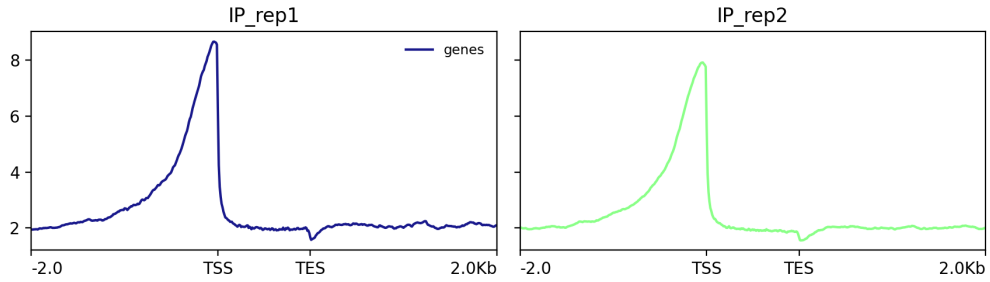
The sequencing data comprised four samples with read counts ranging from 10.1 million (INPUT_rep2) to 28.6 million (INPUT_rep1) reads. FastQC analysis revealed generally high-quality sequencing data, with mean quality scores consistently above Q30 across all base positions and per-sequence quality scores predominantly in the high-quality range. However, several flags were raised by FastQC: per base sequence content showed warnings for all samples, indicating non-uniform base composition in the first ~15 bp of reads, which is typical for random-primed libraries. Per sequence GC content displayed bimodal distributions in all samples, with failures flagged for INPUT samples, suggesting potential contamination or the presence of distinct

sequence populations. Sequence duplication levels were high, particularly in the IP samples (74.3% and 89.1% duplicates in IP_rep2 and IP_rep1, respectively), which is expected for ChIP-seq experiments due to genuine biological enrichment at binding sites. Adapter contamination was detected, with overrepresented sequences identified as Illumina adapters (GATCGGAAGAGCACACGTCTGAACTCCAGTCAC…) comprising 0.6-0.9% of reads in INPUT samples, though adapter content began accumulating only after position 70 bp and reached a maximum of ~5% by the read end.

Following Trimmomatic adapter trimming, alignment rates to the reference genome were excellent, with 96.6-98.9% of reads successfully mapping (INPUT_rep1: 28.6M reads with 1.1% dropped; INPUT_rep2: 10.1M reads with 1.4% dropped; IP_rep1: 27.8M reads with 3.4% dropped; IP_rep2: 28.2M reads with 4.3% dropped). The substantially lower read count in INPUT_rep2 compared to other samples raises some concern about library preparation consistency, though the high mapping rate suggests the data quality itself was not compromised. Overall, the experiment was of high quality and suitable for downstream ChIP-seq analysis. The high duplication rates in IP samples are expected and appropriate for peak calling, the adapter contamination was minimal and successfully removed by trimming, and the excellent alignment rates confirm compatibility with the reference genome. The bimodal GC content distribution, while flagged by FastQC, is not uncommon in ChIP-seq data where enriched regions may have different sequence composition than background. No additional quality improvement steps are necessary for this dataset.

Signal Coverage Plot:

```
img <- image_read("signal_coverage.png")
plot(img)
```



The signal coverage plot displays the aggregate ChIP-seq signal enrichment across all genes, with the x-axis representing genomic positions relative to the transcription start site (TSS) and transcription end site (TES), spanning from 2 kb upstream of the TSS to 2 kb downstream of the TES. The y-axis indicates the normalized read coverage or signal intensity. Both biological replicates (IP_rep1 in blue and IP_rep2 in green) show highly concordant profiles, with a pronounced peak of enrichment centered at the TSS. The signal gradually increases approaching the TSS from the upstream region, reaches a maximum at or immediately downstream of the TSS, and then rapidly declines to baseline levels shortly after transcription initiation. Signal remains relatively flat and low across gene bodies (between TSS and TES) and downstream regions, with minimal enrichment observed beyond the promoter-proximal region.

This enrichment pattern is consistent with RUNX1's known role as a transcription factor that binds primarily to promoter regions to regulate gene expression. As demonstrated by Barutcu et al. (2016), RUNX1 functions prominently in transcriptional activation, with the strongest binding signal observed at promoter regions. The sharp peak at the TSS suggests that RUNX1 preferentially binds to promoters and is directly involved in regulating transcriptional initiation rather than elongation. The lack of sustained signal across gene bodies indicates that RUNX1 does not travel with elongating RNA polymerase II but remains localized to promoter-proximal regions where it can recruit cofactors and chromatin modifiers. This promoter-centric binding pattern supports RUNX1's role in establishing competency for transcription by organizing local chromatin architecture and facilitating enhancer-promoter interactions. The strong reproducibility between

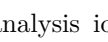
replicates and the clear biological pattern confirm high-quality data suitable for downstream peak calling and analysis of RUNX1-mediated gene regulation in the cellular context being studied.RetryClaude can make mistakes. Please double-check responses.

Motif Finding:

```
img <- image_read("motifs.png")
plot(img)
```

Homer Known Motif Enrichment Results (motifs)

Homer *de novo* Motif Results
Gene Ontology Enrichment Results
Known Motif Enrichment Results (txt file)
Total Target Sequences = 6429, Total Background Sequences = 39982

Rank	Motif	Name	P-value	log P-pvalue	q-value (Benjamini)	# Target Sequences with Motif	% of Targets Sequences with Motif	# Background Sequences with Motif	% of Background Sequences with Motif	Motif File	SVG
1		RUNX(Runt)/HPC7-Runx1-ChIP-Seq(GSE22178)/Homer	1e-650	-1.497e+03	0.0000	1649.0	25.65%	2047.1	5.12%	motif file (matrix)	svg
2		RUNX1(Runt)/Jurkat-RUNX1-ChIP-Seq(GSE29180)/Homer	1e-499	-1.150e+03	0.0000	1739.0	27.05%	2939.7	7.35%	motif file (matrix)	svg
3		RUNX2(Runt)/PCa-RUNX2-ChIP-Seq(GSE33889)/Homer	1e-303	-6.999e+02	0.0000	1263.0	19.65%	2380.4	5.95%	motif file (matrix)	svg
4		RUNX-AML(Runt)/CD4+-PolII-ChIP-Seq(Barski_et_al.)/Homer	1e-303	-6.998e+02	0.0000	1182.0	18.39%	2107.6	5.27%	motif file (matrix)	svg
5		Fosl2(bZIP)/3T3L1-Fosl2-ChIP-Seq(GSE56872)/Homer	1e-180	-4.151e+02	0.0000	519.0	8.07%	682.7	1.71%	motif file (matrix)	svg
6		Fra2(bZIP)/Striatum-Fra2-ChIP-Seq(GSE43429)/Homer	1e-173	-3.989e+02	0.0000	610.0	9.49%	970.1	2.42%	motif file (matrix)	svg
7		GRHL2(CP2)/HBE-GRHL2-ChIP-Seq(GSE46194)/Homer	1e-171	-3.951e+02	0.0000	551.0	8.57%	805.5	2.01%	motif file (matrix)	svg
8		Fra1(bZIP)/BT549-Fra1-ChIP-Seq(GSE46166)/Homer	1e-169	-3.914e+02	0.0000	632.0	9.83%	1053.0	2.63%	motif file (matrix)	svg
9		Fos(bZIP)/TSC-Fos-ChIP-Seq(GSE110950)/Homer	1e-168	-3.889e+02	0.0000	651.0	10.13%	1118.4	2.80%	motif file (matrix)	svg
10		JunB(bZIP)/DendriticCells-JunB-ChIP-Seq(GSE36099)/Homer	1e-163	-3.757e+02	0.0000	628.0	9.77%	1076.6	2.69%	motif file (matrix)	svg
11		BATF(bZIP)/Th17-BATF-ChIP-Seq(GSE39756)/Homer	1e-158	-3.659e+02	0.0000	671.0	10.44%	1238.8	3.10%	motif file (matrix)	svg
12		Atf3(bZIP)/GBM-ATF3-ChIP-Seq(GSE33912)/Homer	1e-154	-3.557e+02	0.0000	684.0	10.64%	1308.2	3.27%	motif file (matrix)	svg
13		Jun-AP1(bZIP)/K562-cJun-ChIP-Seq(GSE31477)/Homer	1e-153	-3.534e+02	0.0000	415.0	6.46%	509.6	1.27%	motif file (matrix)	svg
14		AP-1(bZIP)/ThioMac-PU.1-ChIP-Seq(GSE21512)/Homer	1e-143	-3.299e+02	0.0000	731.0	11.37%	1545.6	3.86%	motif file (matrix)	svg
..		Foxa2(Forkhead)/Liver-Foxa2-ChIP-	1e-							motif file	

The de novo motif discovery analysis identified several significantly enriched transcription factor binding motifs within the ChIP-seq peaks. The top three enriched motifs correspond to RUNX family members (RUNX1, RUNX2) and related RUNX-AML fusion proteins, with RUNX(Runt)/HPC7-Runx1 being the most significantly enriched (p-value = 1e-650, present in 25.65% of target sequences). This high enrichment of RUNX motifs validates the specificity of the ChIP-seq experiment and confirms successful immunoprecipitation of RUNX1-bound regions.

Beyond the expected RUNX motifs, several additional transcription factor motifs were identified, including Fosl2, Fra2, GRHL2, Fra1, Fos, JunB, BATF, Atf3, Jun-AP1, AP-1, and Foxa2. These findings are particularly interesting as they suggest RUNX1 may function as part of larger transcriptional regulatory complexes. Notably, the enrichment of AP-1 family members (Fos, Jun, Fra) is consistent with the Barutcu et al. study, which identified AP1 (FOS/JUN) binding motifs in their RUNX1 ChIP-seq analysis and suggested these factors may be functionally related co-regulators. The presence of multiple bZIP family transcription factors (Fosl2, Fra1, Fra2, BATF, Atf3) indicates RUNX1 may cooperate with these factors to regulate genes involved in cellular differentiation, proliferation, and stress responses. This co-occupancy pattern supports the model that RUNX1 functions not in isolation but as part of combinatorial transcription factor networks that fine-tune gene expression in response to developmental and environmental signals.

Overlap your ChIPseq results with the original RNAseq data:

1. Create a figure that displays the same information of figure 2F from the original publication using your annotated peaks and the RNAseq results. The figure does not have to be the same style but must convey the same information using your results.

Load the data:

```
rnaseq_df <- read_tsv("GSE75070_MCF7_shRUNX1_shNS_RNAseq_log2_foldchange.txt")
```

```
## Rows: 15434 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): genename, transcript
## dbl (2): log2FoldChange, padj
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
peak_df <- read_tsv("results/homer_annotated_full/annotated_peaks.txt")
```

```
## Rows: 6429 Columns: 19
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (10): PeakID (cmd=annotatePeaks.pl repr_peaks_filtered.bed GRCh38.primar...
## dbl  (4): Start, End, Peak Score, Distance to TSS
## lgl  (5): Focus Ratio/Region Size, Nearest Refseq, Nearest Ensembl, Gene Ali...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Sanity check:

```
cat("=== RNA-seq Data ===\n")
```

```
## === RNA-seq Data ===
```

```
cat("Shape:", nrow(rnaseq_df), "rows x", ncol(rnaseq_df), "columns\n")
```

```
## Shape: 15434 rows x 4 columns
```

```
print(head(rnaseq_df))
```

```
## # A tibble: 6 x 4
##   genename transcript                                  log2FoldChange   padj
##   <chr>    <chr>                                               <dbl>  <dbl>
## 1 OARD1    NM_145063                                           0.187  0.546
## 2 WBSCR22  NM_001202560,NM_017528,NR_037776,NR_045512          0.0379 0.907
## 3 PIGZ     NM_025163                                           0.122  0.841
## 4 PRDX5    NM_012094,NM_181651,NM_181652                       0.250  0.118
## 5 PEX16    NM_004813,NM_057174                                -0.301  0.184
## 6 SLC35G2  NM_001097599,NM_001097600,NM_025246                 0.397  NA
```

```r
cat("=== Annotated Peaks Data ===\n")
```

```
## === Annotated Peaks Data ===
```

```r
cat("shape:", nrow(peak_df), "rows x", ncol(peak_df), "columns\n")
```

```
## shape: 6429 rows x 19 columns
```

```r
print(head(peak_df))
```

```
## # A tibble: 6 x 19
##   PeakID (cmd=annotatePeaks.pl repr_pe~1 Chr    Start    End Strand `Peak Score`
##   <chr>                                  <chr>  <dbl>  <dbl> <chr>         <dbl>
## 1 chr11-1015                             chr11 1.18e8 1.18e8 +                 1
## 2 chr12-2110                             chr12 1.18e8 1.18e8 +                 1
## 3 chr15-258                              chr15 4.06e7 4.06e7 +                 1
## 4 chr12-1957                             chr12 7.17e7 7.17e7 +                 1
## 5 chr7-102                               chr7  1.12e8 1.12e8 +                 1
## 6 chr10-40                               chr10 5.49e6 5.49e6 +                 1
## # i abbreviated name:
## #   1: `PeakID (cmd=annotatePeaks.pl repr_peaks_filtered.bed GRCh38.primary_assembly.genome.fa -gtf g
## # i 13 more variables: `Focus Ratio/Region Size` <lgl>, Annotation <chr>,
## #   `Detailed Annotation` <chr>, `Distance to TSS` <dbl>,
## #   `Nearest PromoterID` <chr>, `Entrez ID` <chr>, `Nearest Unigene` <chr>,
## #   `Nearest Refseq` <lgl>, `Nearest Ensembl` <lgl>, `Gene Name` <chr>,
## #   `Gene Alias` <lgl>, `Gene Description` <lgl>, `Gene Type` <chr>
```

Filter the paper's RNA-seq data based on their established thresholds:

```r
sigrna <- rnaseq_df %>% filter(padj < 0.01 & abs(log2FoldChange) > 1)

cat("\n=== Significant DE genes ===\n")
```

```
##
## === Significant DE genes ===
```

```r
cat("Total genes in RNA-seq", nrow(rnaseq_df), "\n")
```

```
## Total genes in RNA-seq 15434
```

```r
cat("Significant genes in RNA_seq", nrow(sigrna), "\n")
```

```
## Significant genes in RNA_seq 1153
```

Find overlapping genes between our ChIP-seq results and their significant RNA-seq data:

```r
sig_de_genes <- sigrna %>% pull(genename) %>% unique() %>% na.omit()

peak_genes <- peak_df %>% pull(`Gene Name`) %>% unique() %>% na.omit()

overlap_genes <- intersect(sig_de_genes, peak_genes)
```

```r
cat("\n=== Overlap Analysis ===\n")
```

```
##
## === Overlap Analysis ===
```

```r
cat("Significant DE genes:", length(sig_de_genes), "\n")
```

```
## Significant DE genes: 1153
```

```r
cat("Genes with ChIP-seq peaks:", length(peak_genes), "\n")
```

```
## Genes with ChIP-seq peaks: 5468
```

```r
cat("Overlapping genes:", length(overlap_genes), "\n")
```

```
## Overlapping genes: 219
```

Create a dataframe of the overlapping genes for further analysis:

```r
overlap_rnaseq <- sigrna %>% filter(genename %in% overlap_genes)

overlap_peaks <- peak_df %>% filter(`Gene Name` %in% overlap_genes)

cat("\nOverlapping RNA-seq entries:", nrow(overlap_rnaseq), "\n")
```

```
##
## Overlapping RNA-seq entries: 219
```

```r
cat("Overlapping peak entries:", nrow(overlap_peaks), "\n")
```

```
## Overlapping peak entries: 258
```

```r
# Preview overlapping genes
print(head(overlap_rnaseq))
```

```
## # A tibble: 6 x 4
##   genename transcript                          log2FoldChange     padj
##   <chr>    <chr>                                       <dbl>    <dbl>
## 1 INCENP   NM_001040694,NM_020238                       1.64 3.30e-14
## 2 TM2D1    NM_032027                                   -1.12 2.42e- 7
## 3 PHGDH    NM_006623                                    2.60 2.98e-30
## 4 DSCC1    NM_024094                                    1.30 6.72e- 9
## 5 IQSEC1   NM_001134382,NM_014869                       1.15 1.16e- 9
## 6 UGDH     NM_001184700,NM_001184701,NM_003359          1.49 4.41e-30
```

Checking which genes have multiple peaks:

```r
overlap_peaks %>%
  count(`Gene Name`) %>%
  filter(n > 1)
```

```
## # A tibble: 35 x 2
##    `Gene Name`     n
##    <chr>       <int>
##  1 ACOXL           2
```

```
##  2 ADRA2C          2
##  3 ALDOA           2
##  4 ANKS3           2
##  5 ARID1A          2
##  6 BCOR            2
##  7 CACNA1D         3
##  8 CNTNAP2         2
##  9 COL5A1          2
## 10 EDN1            2
## # i 25 more rows
```

Recreating Figure 2F from the paper:

Step 1: Categorize DE genes as up or down regulated

```
up_genes <- sigrna %>% filter(log2FoldChange > 1) %>% pull(genename) %>% unique()

down_genes <- sigrna %>% filter(log2FoldChange < -1) %>% pull(genename) %>% unique()

cat("Up-regulated genes:", length(up_genes), "\n")
```

```
## Up-regulated genes: 687
```

```
cat("Down-regulated genes:", length(down_genes), "\n")
```

```
## Down-regulated genes: 466
```

Loading GTF file to create +/-kb TSS of whole gene body:

```
# Read the GTF file to get GENE coordinates
gtf <- rtracklayer::import("refs/gencode.v45.primary_assembly.annotation.gtf")
genes_gr <- gtf[gtf$type == "gene"]
```

```
# Create gene coordinate lookup table
gene_coords <- data.frame(
  gene_name = genes_gr$gene_name,
  chr = as.character(seqnames(genes_gr)),
  gene_start = start(genes_gr),
  gene_end = end(genes_gr),
  stringsAsFactors = FALSE
) %>%
  distinct(gene_name, .keep_all = TRUE)

# Remove any existing gene coordinate columns from peak_df
peak_df_clean <- peak_df %>%
  select(-any_of(c("gene_start", "gene_end", "chr", "gene_start.x", "gene_start.y",
                   "gene_end.x", "gene_end.y", "gene_coords")))

# Now perform the merge
peak_df_with_coords <- peak_df_clean %>%
  left_join(gene_coords, by = c("Gene Name" = "gene_name"))

# Check the merge
```

```r
cat("Rows with gene coordinates:", sum(!is.na(peak_df_with_coords$gene_start)), "\n")
```

```
## Rows with gene coordinates: 6428
```

Step 2: Get genes with RUNX1 peaks at different distances

```r
# +/- 5kb of TSS
genes_with_peaks_5kb <- peak_df %>%
  filter(abs(`Distance to TSS`) <= 5000) %>%
  pull(`Gene Name`) %>%
  unique()

# +/- 20kb of whole gene
genes_with_peaks_20kb_gene <- peak_df_with_coords %>%
  filter(
    !is.na(gene_start),
    !is.na(gene_end),
    Chr == chr,
    End >= (gene_start - 20000),
    Start <= (gene_end + 20000)
  ) %>%
  pull(`Gene Name`) %>%
  unique()

cat("Genes with peaks within 5kb of TSS:", length(genes_with_peaks_5kb), "\n")
```

```
## Genes with peaks within 5kb of TSS: 3962
```

```r
cat("Genes with peaks within 20kb of whole gene:", length(genes_with_peaks_20kb_gene), "\n")
```

```
## Genes with peaks within 20kb of whole gene: 5114
```

Step 3: Calculate overlap for each category and distance

```r
# 5kb TSS
up_bound_5kb <- sum(up_genes %in% genes_with_peaks_5kb)

up_not_bound_5kb <- length(up_genes) - up_bound_5kb

down_bound_5kb <- sum(down_genes %in% genes_with_peaks_5kb)

down_not_bound_5kb <- length(down_genes) - down_bound_5kb

# 20kb
up_bound_20kb <- sum(up_genes %in% genes_with_peaks_20kb_gene)

up_not_bound_20kb <- length(up_genes) - up_bound_20kb

down_bound_20kb <- sum(down_genes %in% genes_with_peaks_20kb_gene)

down_not_bound_20kb <- length(down_genes) - down_bound_20kb
```

Step 4: Create dataframe for plotting

```
plot_data <- data.frame(
  Category = c("Up-regulated", "Up-regulated", "Down-regulated", "Down-regulated",
               "Up-regulated", "Up-regulated", "Down-regulated", "Down-regulated"),
  Binding = c("RUNX1 bound", "Not bound", "RUNX1 bound", "Not bound",
              "RUNX1 bound", "Not bound", "RUNX1 bound", "Not bound"),
  Count = c(up_bound_5kb, up_not_bound_5kb, down_bound_5kb, down_not_bound_5kb,
            up_bound_20kb, up_not_bound_20kb, down_bound_20kb, down_not_bound_20kb),
  Distance = c(rep("+/- 5kb of TSS", 4), rep("+/- 20kb of whole gene", 4))
)
plot_data
```

```
##         Category     Binding Count               Distance
## 1   Up-regulated RUNX1 bound    97         +/- 5kb of TSS
## 2   Up-regulated   Not bound   590         +/- 5kb of TSS
## 3 Down-regulated RUNX1 bound    71         +/- 5kb of TSS
## 4 Down-regulated   Not bound   395         +/- 5kb of TSS
## 5   Up-regulated RUNX1 bound   123 +/- 20kb of whole gene
## 6   Up-regulated   Not bound   564 +/- 20kb of whole gene
## 7 Down-regulated RUNX1 bound    86 +/- 20kb of whole gene
## 8 Down-regulated   Not bound   380 +/- 20kb of whole gene
```

```
# Calculate percentages within each category and distance
plot_data <- plot_data %>% group_by(Category, Distance) %>% mutate(Percentage = Count / sum(Count) * 10
plot_data
```

```
## # A tibble: 8 x 5
##   Category       Binding     Count Distance               Percentage
##   <chr>          <chr>       <int> <chr>                       <dbl>
## 1 Up-regulated   RUNX1 bound    97 +/- 5kb of TSS               14.1
## 2 Up-regulated   Not bound     590 +/- 5kb of TSS               85.9
## 3 Down-regulated RUNX1 bound    71 +/- 5kb of TSS               15.2
## 4 Down-regulated Not bound     395 +/- 5kb of TSS               84.8
## 5 Up-regulated   RUNX1 bound   123 +/- 20kb of whole gene       17.9
## 6 Up-regulated   Not bound     564 +/- 20kb of whole gene       82.1
## 7 Down-regulated RUNX1 bound    86 +/- 20kb of whole gene       18.5
## 8 Down-regulated Not bound     380 +/- 20kb of whole gene       81.5
```

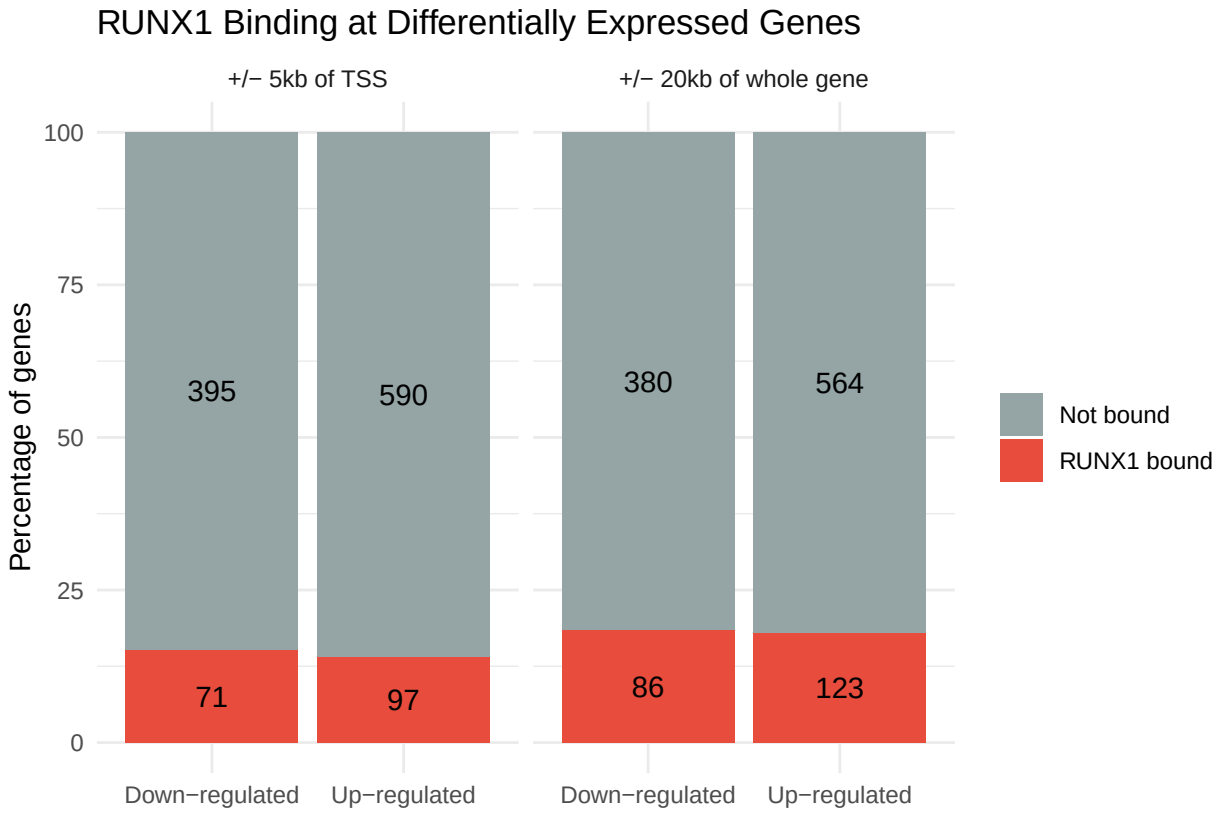
```
# Set factor order for proper display
plot_data$Binding <- factor(plot_data$Binding, levels = c("Not bound", "RUNX1 bound"))
plot_data$Distance <- factor(plot_data$Distance, levels = c("+/- 5kb of TSS", "+/- 20kb of whole gene"))

print(plot_data)
```

```
## # A tibble: 8 x 5
##   Category       Binding     Count Distance               Percentage
##   <chr>          <fct>       <int> <fct>                       <dbl>
## 1 Up-regulated   RUNX1 bound    97 +/- 5kb of TSS               14.1
## 2 Up-regulated   Not bound     590 +/- 5kb of TSS               85.9
## 3 Down-regulated RUNX1 bound    71 +/- 5kb of TSS               15.2
## 4 Down-regulated Not bound     395 +/- 5kb of TSS               84.8
## 5 Up-regulated   RUNX1 bound   123 +/- 20kb of whole gene       17.9
## 6 Up-regulated   Not bound     564 +/- 20kb of whole gene       82.1
## 7 Down-regulated RUNX1 bound    86 +/- 20kb of whole gene       18.5
```

```
## 8 Down-regulated Not bound     380 +/- 20kb of whole gene       81.5
```

Step 5: Create stacked bar chart (side by side like Figure 2F)

```
ggplot(plot_data, aes(x = Category, y = Percentage, fill = Binding)) +
  geom_bar(stat = "identity", position = "stack") +
  geom_text(aes(label = Count),
            position = position_stack(vjust = 0.5),
            size = 4) +
  scale_fill_manual(values = c("Not bound" = "#95A5A6", "RUNX1 bound" = "#E74C3C")) +
  facet_wrap(~Distance) +
  labs(x = "",
       y = "Percentage of genes",
       title = "RUNX1 Binding at Differentially Expressed Genes") +
  theme_minimal() +
  theme(legend.title = element_blank(), legend.position = "right")
```



Do you observe any differences in the number of overlapping genes from both analyses? - If you do observe a difference, explain at least two factors that may have contributed to these differences.

Yes, our results differ from the paper's Figure F, likely because:

1. Genome version (hg19 vs hg38): The original study used hg19 while we used hg38. Gene annotations and TSS positions shifted between assemblies.

For the +/- 5kb of TSS analysis, the paper identified 245 total RUNX1-bound differentially expressed genes (84 down-regulated, 161 up-regulated), while we identified 168 genes (71 down-regulated, 97 up-regulated). This represents 77 fewer genes (31% reduction) in our analysis. For the +/- 20kb of whole gene analysis, the paper reported 289 RUNX1-bound genes (128 down-regulated, 161 up-regulated), compared to my 209 genes (86 down-regulated, 123 up-regulated), a difference of 80 genes (28% reduction).

The largest discrepancy occurs in up-regulated genes within 5kb of TSS, where we detected 64 fewer genes than the paper (97 vs 161, representing a 40% reduction). Despite these quantitative differences, the overall biological trends remain consistent: both analyses show that the 20kb whole gene approach captures more RUNX1-bound genes than the 5kb TSS approach, and up-regulated genes exhibit more RUNX1 binding than down-regulated genes across both distance windows.

2. Different peak reproducibility and filtering criteria: Both analyses used HOMER for peak calling, but the downstream reproducibility and filtering steps likely differed because the paper did not specify how they defined "reproducible peaks." In our pipeline, we required direct overlap between peaks from both replicates and removed ENCODE blacklist regions, whereas the paper may have used alternative methods such as IDR thresholds, different overlap criteria, or different blacklist usage. These stricter filtering steps in our workflow likely reduced the number of retained peaks, leading to fewer RUNX1-bound differentially expressed genes compared to the paper.
3. INPUT_rep2 has much lower read depth than the other samples, which can reduce peak-calling sensitivity and may have contributed to NEAT1 not being detected.

What is the rationale behind combining these two analyses in this way? What additional conclusions is it supposed to enable you to draw?

The rationale behind combining the 5kb TSS and 20kb whole gene analyses is to distinguish between different modes of RUNX1-mediated transcriptional regulation and to assess the relative contributions of promoter-proximal versus distal regulatory mechanisms. The 5kb TSS window captures RUNX1 binding at promoters, representing direct transcriptional control, while the 20kb whole gene window includes enhancers, intronic regulatory elements, and other distal sites that can regulate gene expression through long-range chromatin interactions. Comparing these two approaches reveals that a substantial portion of RUNX1 regulation occurs through distal elements. In our analysis, 24% more genes (41 additional genes) were captured with the broader window, demonstrating that enhancer-mediated regulation is an important component of RUNX1's function. This dual approach also enables us to distinguish directly regulated genes (those with RUNX1 binding) from indirectly affected genes (differentially expressed without nearby RUNX1 binding), revealing that RUNX1 functions within broader regulatory networks where direct binding to some genes triggers cascading effects throughout the transcriptome. Overall, combining these analyses provides insight into both the mechanisms and the scope of RUNX1's role in controlling gene expression programs.

2. In figures 2D and 2E, the authors identify and highlight two specific genes that were identified in both experiments. Using your list of filtered and reproducible peaks, a genome browser of your choice, and your bigWig files, please re-create these figures with your own results (You do not need to include the RNAseq data, but you should re-create the genomic tracks from your ChIPseq results)

```
# Check if these genes are in your overlapping genes
"MALAT1" %in% overlap_genes
```

```
## [1] TRUE
```

```
"NEAT1" %in% overlap_genes
```

```
## [1] FALSE
```

```
# Check MALAT1 in your filtered peaks
peak_df %>%
  filter(`Gene Name` == "MALAT1")
```

```
## # A tibble: 2 x 19
##   PeakID (cmd=annotatePeaks.pl repr_pe~1 Chr    Start    End Strand `Peak Score`
##   <chr>                                  <chr>  <dbl>  <dbl> <chr>         <dbl>
## 1 chr11-162                              chr11 6.55e7 6.55e7 +                 1
## 2 chr11-11                               chr11 6.55e7 6.55e7 +                 1
## # i abbreviated name:
## #   1: `PeakID (cmd=annotatePeaks.pl repr_peaks_filtered.bed GRCh38.primary_assembly.genome.fa -gtf g
## # i 13 more variables: `Focus Ratio/Region Size` <lgl>, Annotation <chr>,
## #   `Detailed Annotation` <chr>, `Distance to TSS` <dbl>,
## #   `Nearest PromoterID` <chr>, `Entrez ID` <chr>, `Nearest Unigene` <chr>,
## #   `Nearest Refseq` <lgl>, `Nearest Ensembl` <lgl>, `Gene Name` <chr>,
## #   `Gene Alias` <lgl>, `Gene Description` <lgl>, `Gene Type` <chr>
```

```
# Look up NEAT1 in the RNA-seq data
rnaseq_df %>%
  filter(genename == "NEAT1")
```

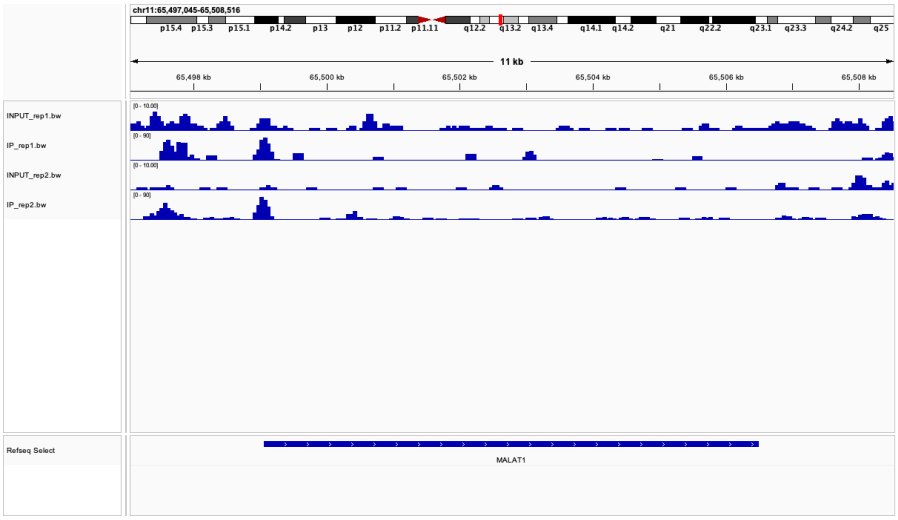
```
## # A tibble: 1 x 4
##   genename transcript log2FoldChange     padj
##   <chr>    <chr>               <dbl>    <dbl>
## 1 NEAT1    NR_028272           -1.95 1.97e-41
```

```
# Check if NEAT1 has peaks
peak_df %>%
  filter(`Gene Name` == "NEAT1")
```

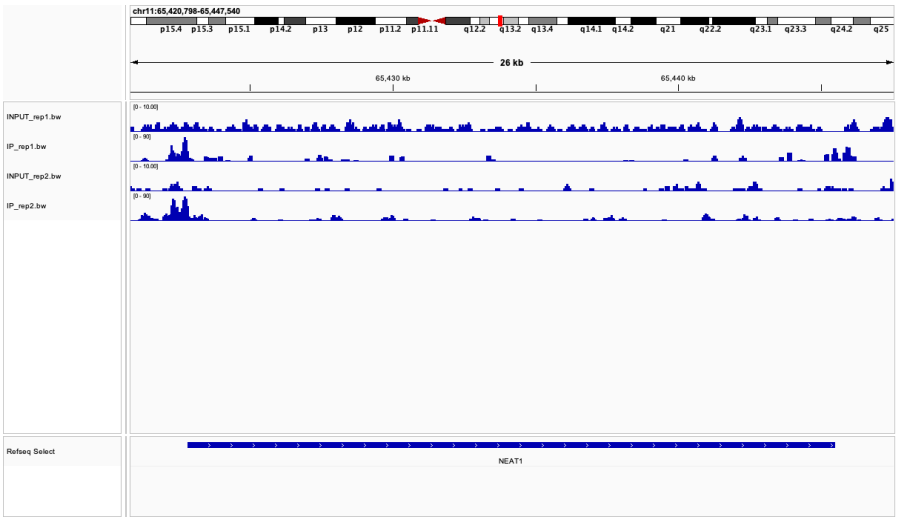
```
## # A tibble: 0 x 19
## # i 19 variables:
## #   PeakID (cmd=annotatePeaks.pl repr_peaks_filtered.bed GRCh38.primary_assembly.genome.fa -gtf genc
## #   Chr <chr>, Start <dbl>, End <dbl>, Strand <chr>, Peak Score <dbl>,
## #   Focus Ratio/Region Size <lgl>, Annotation <chr>, Detailed Annotation <chr>,
## #   Distance to TSS <dbl>, Nearest PromoterID <chr>, Entrez ID <chr>,
## #   Nearest Unigene <chr>, Nearest Refseq <lgl>, Nearest Ensembl <lgl>,
```

```
## #   Gene Name <chr>, Gene Alias <lgl>, Gene Description <lgl>, ...
```

```
img <- image_read("malat1.png")
plot(img)
```




```
img <- image_read("neat1.png")
plot(img)
```



From your annotated peaks, do you observe statistically significant peaks in these same two genes? - How similar do your genomic tracks appear to those in the paper? If you observe any differences, comment briefly on why there may be discrepancies.

From our annotated peaks, we observe a statistically significant RUNX1 peak at MALAT1, consistent with the original paper's findings. However, NEAT1 does not appear in our final annotated peaks file, representing a key discrepancy from the published results. Investigation revealed that NEAT1 had a significant peak called in replicate 2 (chr11:65,422,232-65,422,470) but not in replicate 1, causing it to fail our reproducibility filter that required peaks in both replicates. Our genomic tracks for MALAT1 appear similar to those in the paper, showing clear RUNX1 binding signal above input background, while NEAT1 shows visual ChIP-seq

signal in the bigWig files that was nonetheless filtered out. These discrepancies likely stem from differences in genome version (hg38 vs hg19), and stringency of reproducibility requirements between our analysis and the original study.

```
# Check if these genes are in your overlapping genes
"MALAT1" %in% overlap_genes
```

```
## [1] TRUE
```

```
"NEAT1" %in% overlap_genes
```

```
## [1] FALSE
```

Re-create the table found in supplementary figure S2A. Compare the results with your own findings.

```
# For gzipped FASTQ files
count_fastq_reads <- function(file) {
  # Read all lines
  lines <- readLines(gzfile(file))
  # Each read = 4 lines
  num_reads <- length(lines) / 4
  return(num_reads)
}

# Count reads in each file
ip_rep1_reads <- count_fastq_reads("/projectnb/bf528/materials/project-3-chipseq/full_files/IP_rep1.fas
ip_rep2_reads <- count_fastq_reads("/projectnb/bf528/materials/project-3-chipseq/full_files/IP_rep2.fas
input_rep1_reads <- count_fastq_reads("/projectnb/bf528/materials/project-3-chipseq/full_files/INPUT_re
input_rep2_reads <- count_fastq_reads("/projectnb/bf528/materials/project-3-chipseq/full_files/INPUT_re

cat("IP rep1:", ip_rep1_reads, "\n")
```

```
## IP rep1: 29734121
```

```
cat("IP rep2:", ip_rep2_reads, "\n")
```

```
## IP rep2: 29988988
```

```
cat("INPUT rep1:", input_rep1_reads, "\n")
```

```
## INPUT rep1: 30075142
```

```
cat("INPUT rep2:", input_rep2_reads, "\n")
```

```
## INPUT rep2: 10900442
```

```
# Read flagstat file and extract mapped reads
extract_mapped <- function(flagstat_file) {
  lines <- readLines(flagstat_file)
  mapped_line <- grep("mapped", lines, value = TRUE)
  # Extract first number
  mapped_count <- as.numeric(sub(" .*", "", mapped_line))
  return(mapped_count)
}

# Use it
```

```r
ip_rep1_mapped <- extract_mapped("results/flagstat/IP_rep1.flagstat.txt")
ip_rep2_mapped <- extract_mapped("results/flagstat/IP_rep2.flagstat.txt")
input_rep1_mapped <- extract_mapped("results/flagstat/INPUT_rep1.flagstat.txt")
input_rep2_mapped <- extract_mapped("results/flagstat/INPUT_rep2.flagstat.txt")

cat("IP rep1 mapped:", ip_rep1_mapped, "\n")
```

```
## IP rep1 mapped: 27788333 27788333 0 0 0
```

```r
cat("IP rep2 mapped:", ip_rep2_mapped, "\n")
```

```
## IP rep2 mapped: 28197249 28197249 0 0 0
```

```r
cat("INPUT rep1 mapped:", input_rep1_mapped, "\n")
```

```
## INPUT rep1 mapped: 28578769 28578769 0 0 0
```

```r
cat("INPUT rep2 mapped:", input_rep2_mapped, "\n")
```

```
## INPUT rep2 mapped: 10055226 10055226 0 0 0
```

```r
# Create the table with your actual data
table_a <- data.frame(
  `Sample Name` = c("RUNX1 ChIP 1", "RUNX1 ChIP 2", "INPUT 1", "INPUT 2"),
  `Biological Replicate` = c(1, 2, 1, 2),
  `Raw Reads` = c(29734121, 29988988, 30075142, 10900442),
  `Mapped Reads` = c(27788333, 28197249, 28578769, 10055226),
  check.names = FALSE
)

# Display as a nice table
kable(table_a, format = "markdown",
      caption = "Sequencing and alignment statistics for RUNX1 ChIP-seq")
```

```
## Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 1: Sequencing and alignment statistics for RUNX1 ChIP-seq

Sample Name	Biological Replicate	Raw Reads	Mapped Reads
RUNX1 ChIP 1	1	29734121	27788333
RUNX1 ChIP 2	2	29988988	28197249
INPUT 1	1	30075142	28578769
INPUT 2	2	10900442	10055226

```r
# Or print it
print(table_a)
```

```
##    Sample Name Biological Replicate Raw Reads Mapped Reads
## 1 RUNX1 ChIP 1                    1  29734121     27788333
## 2 RUNX1 ChIP 2                    2  29988988     28197249
```

```
## 3      INPUT 1                   1  30075142     28578769
## 4      INPUT 2                   2  10900442     10055226
```

Do you observe differences in the reported number of raw and mapped reads?

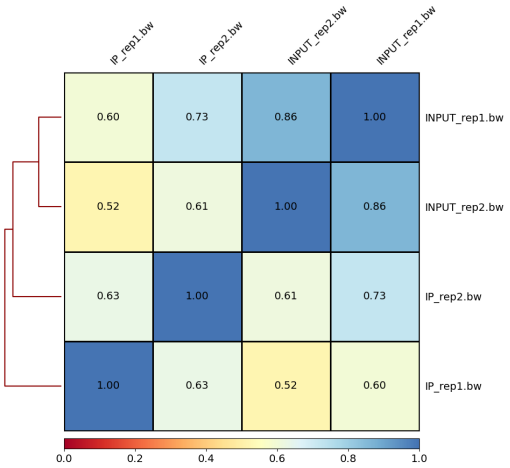
Yes, we observe substantial differences in both raw and mapped read counts between our analysis and the paper.

If so, provide at least two explanations for the discrepancies.

1. Different genome builds: We aligned to GRCh38, while the paper have used an older genome version (hg19).
2. Different filtering steps: While our flagstat counts represent aligned reads after trimming, the paper may have applied additional filtering steps or used different parameters while trimming.

Compare your correlation plot with the one found in supplementary figure S2B.

```
img <- image_read("results/plotcorr_full/correlation_plot.png")
plot(img)
```



Do you observe any differences in your calculated metrics?

Yes, we observe lower correlation values compared to the paper. Most notably, our IP replicate correlation (0.63) is substantially lower than the paper's RUNX1 ChIP replicate correlation (0.91). Our INPUT replicates show reasonably good agreement (0.86 vs 0.95 in paper), but still slightly lower. The correlations between IP and INPUT samples are also generally lower in our analysis (0.52-0.73 range vs 0.69-0.76 in paper). Additionally, it should be noted that the paper used Pearson correlation coefficient while our analysis used Spearman correlation, which could contribute to the observed differences in correlation values.

What was the author's takeaway from this figure? What is your conclusion from this figure regarding the success of the experiment?

The authors' takeaway was that biological replicates showed high correlation (0.91 for ChIP, 0.95 for INPUT), indicating good reproducibility and experimental quality. The lower correlations between ChIP and INPUT samples (0.69-0.76) confirmed appropriate signal enrichment in the immunoprecipitated samples.

Our conclusion is that while our experiment shows moderate reproducibility, there is more variability between replicates than observed in the original study. The IP replicate correlation of 0.63 suggests reasonable but not excellent agreement, which may explain why some peaks (like NEAT1) were called in only one replicate. The use of Spearman correlation (which measures monotonic relationships) versus Pearson correlation (which measures linear relationships) may account for some of the numerical differences, as Spearman is more robust to outliers but may yield lower values for strongly linear relationships. The INPUT replicates show better correlation (0.86), indicating more consistent background signal. The hierarchical clustering correctly groups IP replicates together and INPUT replicates together, confirming that the experiment successfully enriched for RUNX1-bound regions despite the moderate replicate correlation. This lower reproducibility likely contributed to our more stringent filtering and the differences observed in peak calling results.

Create a venn diagram with the same information as found in figure S2C.

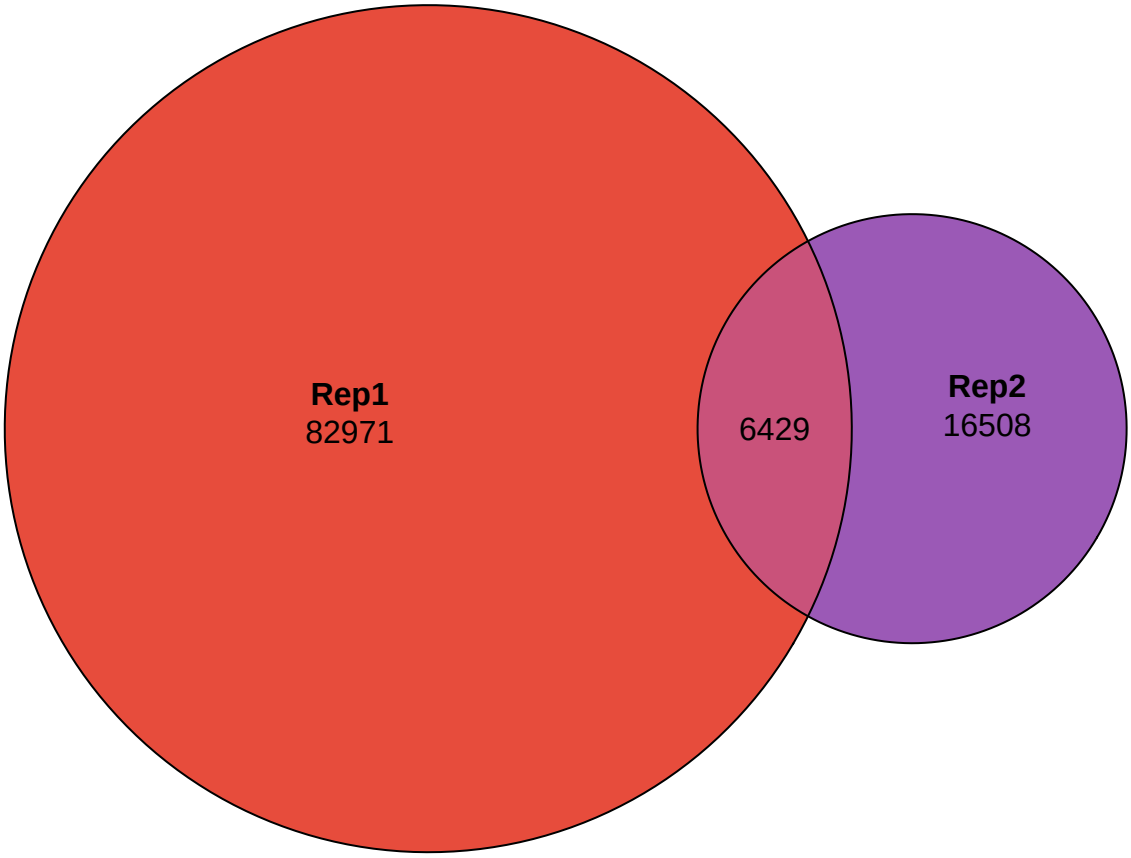
```
# Count peaks in each replicate file
rep1_count <- length(readLines("results/findpeaks_full/rep1_peaks.txt"))
rep2_count <- length(readLines("results/findpeaks_full/rep2_peaks.txt"))

cat("Rep1 peaks:", rep1_count, "\n")
```

```
## Rep1 peaks: 89400
```

```
cat("Rep2 peaks:", rep2_count, "\n")
```

```
## Rep2 peaks: 22937
```

```
fit <- euler(c(
  "Rep1" = 82971,      # Only in rep1 (89400 - 6429)
  "Rep2" = 16508,      # Only in rep2 (22937 - 6429)
  "Rep1&Rep2" = 6429   # Overlap
))

plot(fit,
     fills = c("#E74C3C", "#9B59B6"),
     quantities = TRUE,
     labels = c("Rep1", "Rep2"))
```



Do you observe any differences in your results compared to what you see?

Yes, we observe substantial differences in both the total number of peaks and the distribution between replicates.

If so, provide at least two explanations for the discrepancies in the number of called peaks.

1. Different genome builds: We aligned to GRCh38, while the paper have used an older genome version (hg19).
2. Different post-peak-calling filtering steps: Both analyses used HOMER for peak calling, but the downstream reproducibility and filtering steps likely differed because the paper did not specify how they defined "reproducible peaks." In our pipeline, we required direct overlap between peaks from both replicates and removed ENCODE blacklist regions, whereas the paper may have used alternative methods such as IDR thresholds, different overlap criteria, or different blacklist usage. These stricter filtering steps in our workflow likely reduced the number of retained peaks, leading to fewer RUNX1-bound differentially expressed genes compared to the paper.

Analyze the annotated peaks.

In your created notebook, detail the methodology used to perform the enrichment:

Gene enrichment analysis was performed using Enrichr (https://maayanlab.cloud/Enrichr/). 5468 unique gene names were extracted from our annotated peaks file and were submitted to Enrichr for analysis. The enrichment was analyzed across multiple databases including GO Biological Process 2025, GO Molecular Function 2025, WikiPathway 2024 Human, and KEGG 2021 Human pathways to identify overrepresented biological processes and pathways.

Create a single figure / plot / table that displays some of the top results from the analysis.

```
# Get unique gene names from your peaks
gene_list <- peak_df %>%
  pull(`Gene Name`) %>%
  unique() %>%
  na.omit()

# Check available databases
dbs <- listEnrichrDbs()

# Run enrichment on common databases
dbs_to_use <- c("GO_Biological_Process_2025",
                "GO_Molecular_Function_2025",
                "KEGG_2021_Human",
                "WikiPathway_2024_Human",
                "Reactome_Pathways_2024")

enriched <- enrichr(gene_list, dbs_to_use)
```

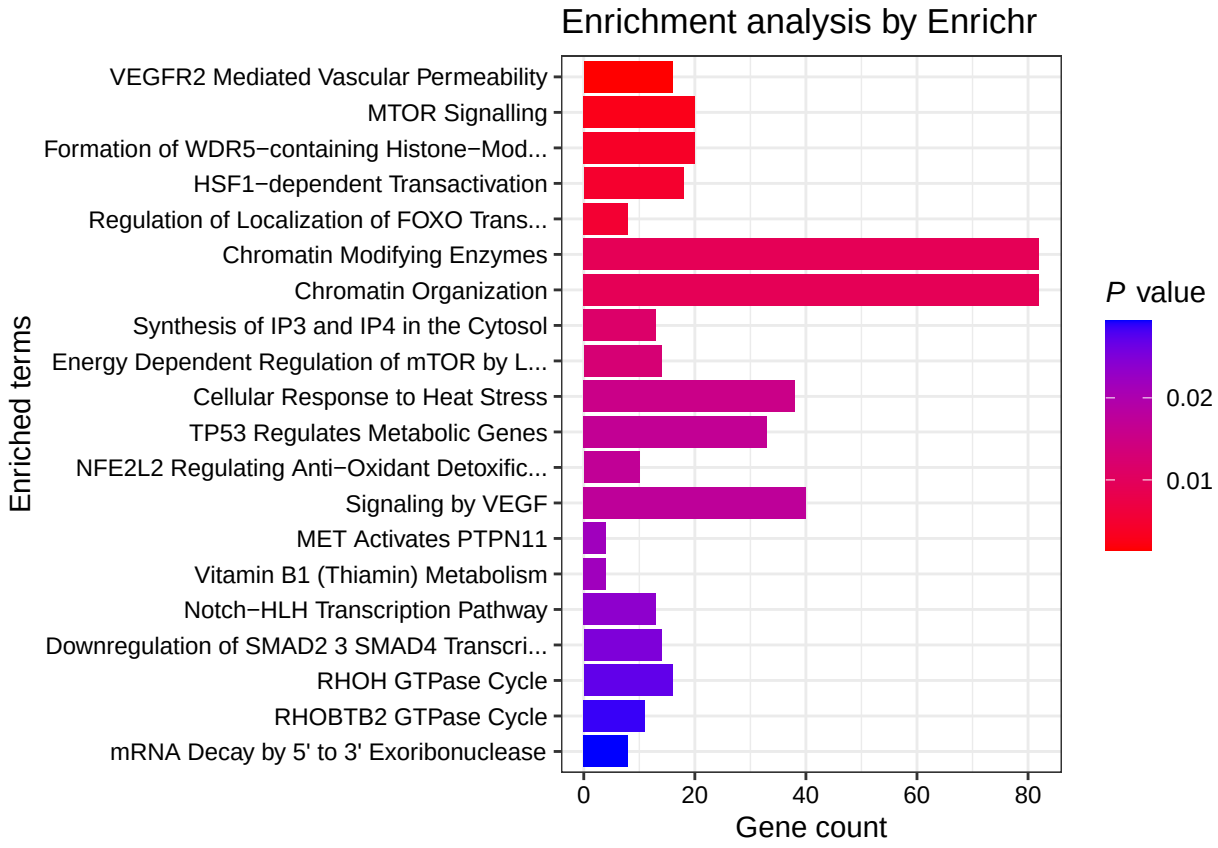
```
## Uploading data to Enrichr... Done.
##   Querying GO_Biological_Process_2025... Done.
##   Querying GO_Molecular_Function_2025... Done.
##   Querying KEGG_2021_Human... Done.
##   Querying WikiPathway_2024_Human... Done.
##   Querying Reactome_Pathways_2024... Done.
## Parsing results... Done.
```

```
# View results
head(enriched$Reactome_Pathways_2024)
```

```
##                                                    Term Overlap     P.value
## 1                    VEGFR2 Mediated Vascular Permeability   16/29 0.001423855
## 2                                       MTOR Signalling   20/41 0.002767811
## 3 Formation of WDR5-containing Histone-Modifying Complexes   20/42 0.003941961
## 4                           HSF1-dependent Transactivation   18/37 0.004603817
## 5 Regulation of Localization of FOXO Transcription Factors    8/12 0.005101657
## 6                             Chromatin Modifying Enzymes  82/238 0.009155912
##   Adjusted.P.value Old.P.value Old.Adjusted.P.value Odds.Ratio Combined.Score
## 1        0.9999962           0                   0   3.277612      21.482737
## 2        0.9999962           0                   0   2.536711      14.940461
```

```
## 3        0.9999962           0                   0    2.421239      13.404165
## 4        0.9999962           0                   0    2.522781      13.574757
## 5        0.9999962           0                   0    5.321612      28.088478
## 6        0.9999962           0                   0    1.403011       6.584828
##
## 1
## 2
## 3
## 4
## 5
## 6 PHF2;KDM5B;KDM5C;KDM1A;KDM1B;EHMT1;CHD4;ACTB;ING3;KAT5;MECOM;RUVBL1;KAT8;KMT5B;PRMT6;NCOA2;PBRM1;PH
```

```
# Plot top results
plotEnrich(enriched$Reactome_Pathways_2024, showTerms = 20, numChar = 40)
```



Comment briefly in a paragraph about the results you observe and why they may be interesting.

The enrichment analysis revealed that genes associated with RUNX1 binding peaks are significantly enriched for processes related to mesenchymal cell proliferation, membrane organization, and DNA-directed DNA polymerase activity (GO Biological Process), as well as chromatin modifying enzymes and chromatin organization (Reactome Pathways). These results are highly consistent with RUNX1's known roles described in the literature. The enrichment for positive regulation of mesenchymal cell proliferation aligns with RUNX1's dual function as both an oncogene and tumor suppressor in breast cancer, particularly in epithelial-mesenchymal transition processes. Most notably, the strong enrichment for chromatin organization and chromatin modifying enzyme pathways directly validates the paper's central finding that RUNX1 functions in cooperation

with chromatin modifier and remodeling enzymes to alter chromatin structure and higher-order genome organization. The presence of DNA polymerase activity and various regulatory processes further supports RUNX1's diverse roles in both promoter and enhancer regulation, consistent with the observation that 30% of RUNX1 binding sites are intergenic, suggesting functions beyond simple transcriptional activation.

```
sessioninfo::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.4.3 (2025-02-28)
##  os       AlmaLinux 8.10 (Cerulean Leopard)
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/New_York
##  date     2025-11-24
##  pandoc   3.2 @ /usr/local/ood/rstudio-server-2024.12.0+467/bin/quarto/bin/tools/x86_64/ (via rmarkd
##  quarto   1.5.57 @ /usr/local/ood/rstudio-server-2024.12.0+467/bin/quarto/bin/quarto
##
## - Packages -------------------------------------------------------------------
##  package              * version   date (UTC) lib source
##  abind                  1.4-8     2024-09-12 [1] CRAN (R 4.4.0)
##  Biobase                2.66.0    2024-10-29 [1] Bioconductor 3.20 (R 4.4.0)
##  BiocGenerics         * 0.52.0    2024-10-29 [1] Bioconductor 3.20 (R 4.4.0)
##  BiocIO                 1.16.0    2024-10-29 [1] Bioconductor 3.20 (R 4.4.0)
##  BiocParallel           1.40.2    2025-10-06 [1] Bioconductor
##  Biostrings             2.74.1    2024-12-16 [1] Bioconductor 3.20 (R 4.4.3)
##  bit                    4.6.0     2025-03-06 [1] CRAN (R 4.4.0)
##  bit64                  4.6.0-1   2025-01-16 [1] CRAN (R 4.4.0)
##  bitops                 1.0-9     2024-10-03 [1] CRAN (R 4.4.0)
##  cli                    3.6.5     2025-04-23 [1] CRAN (R 4.4.0)
##  codetools              0.2-20    2024-03-31 [2] CRAN (R 4.4.3)
##  colorspace             2.1-2     2025-09-22 [1] CRAN (R 4.4.0)
##  crayon                 1.5.3     2024-06-20 [1] CRAN (R 4.4.0)
##  curl                   7.0.0     2025-08-19 [1] CRAN (R 4.4.0)
##  DelayedArray           0.32.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  digest                 0.6.37    2024-08-19 [1] CRAN (R 4.4.0)
##  dplyr                * 1.1.4     2023-11-17 [2] CRAN (R 4.4.3)
##  enrichR              * 3.4       2025-02-02 [2] CRAN (R 4.4.3)
##  eulerr               * 7.0.4     2025-09-24 [1] CRAN (R 4.4.3)
##  evaluate               1.0.5     2025-08-27 [1] CRAN (R 4.4.0)
##  farver                 2.1.2     2024-05-13 [1] CRAN (R 4.4.0)
##  fastmap                1.2.0     2024-05-15 [1] CRAN (R 4.4.0)
##  forcats              * 1.0.0     2023-01-29 [2] CRAN (R 4.4.3)
##  generics               0.1.4     2025-05-09 [1] CRAN (R 4.4.0)
##  GenomeInfoDb         * 1.42.3    2025-01-27 [2] Bioconductor 3.20 (R 4.4.3)
##  GenomeInfoDbData       1.2.13    2025-10-06 [1] Bioconductor
##  GenomicAlignments      1.42.0    2024-10-29 [1] Bioconductor 3.20 (R 4.4.3)
##  GenomicRanges        * 1.58.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  ggplot2              * 3.5.1     2024-04-23 [2] CRAN (R 4.4.3)
##  glue                   1.8.0     2024-09-30 [1] CRAN (R 4.4.0)
##  gtable                 0.3.6     2024-10-25 [2] CRAN (R 4.4.3)
##  hms                    1.1.3     2023-03-21 [2] CRAN (R 4.4.3)
```

```
##  htmltools             0.5.8.1   2024-04-04 [2] CRAN (R 4.4.3)
##  httr                  1.4.7     2023-08-15 [2] CRAN (R 4.4.3)
##  IRanges             * 2.40.1    2024-12-05 [2] Bioconductor 3.20 (R 4.4.3)
##  jsonlite              2.0.0     2025-03-27 [1] CRAN (R 4.4.0)
##  knitr               * 1.49      2024-11-08 [2] CRAN (R 4.4.3)
##  labeling              0.4.3     2023-08-29 [2] CRAN (R 4.4.3)
##  lattice               0.22-7    2025-04-02 [1] CRAN (R 4.4.0)
##  lifecycle             1.0.4     2023-11-07 [2] CRAN (R 4.4.3)
##  lubridate           * 1.9.4     2024-12-08 [2] CRAN (R 4.4.3)
##  magick              * 2.8.5     2024-09-20 [2] CRAN (R 4.4.3)
##  magrittr              2.0.4     2025-09-12 [1] CRAN (R 4.4.0)
##  Matrix                1.7-4     2025-08-28 [2] CRAN (R 4.4.3)
##  MatrixGenerics        1.18.1    2025-01-09 [2] Bioconductor 3.20 (R 4.4.3)
##  matrixStats           1.5.0     2025-01-07 [1] CRAN (R 4.4.0)
##  munsell               0.5.1     2024-04-01 [2] CRAN (R 4.4.3)
##  pillar                1.10.1    2025-01-07 [2] CRAN (R 4.4.3)
##  pkgconfig             2.0.3     2019-09-22 [2] CRAN (R 4.4.3)
##  polyclip              1.10-7    2024-07-23 [1] CRAN (R 4.4.0)
##  polylabelr            0.3.0     2024-11-19 [1] CRAN (R 4.4.3)
##  purrr               * 1.0.4     2025-02-05 [2] CRAN (R 4.4.3)
##  R6                    2.6.1     2025-02-15 [1] CRAN (R 4.4.0)
##  Rcpp                  1.1.0     2025-07-02 [1] CRAN (R 4.4.0)
##  RCurl                 1.98-1.16 2024-07-11 [2] CRAN (R 4.4.3)
##  readr               * 2.1.5     2024-01-10 [2] CRAN (R 4.4.3)
##  restfulr              0.0.16    2025-06-27 [1] CRAN (R 4.4.3)
##  rjson                 0.2.23    2024-09-16 [1] CRAN (R 4.4.0)
##  rlang                 1.1.6     2025-04-11 [1] CRAN (R 4.4.0)
##  rmarkdown             2.29      2024-11-04 [2] CRAN (R 4.4.3)
##  Rsamtools             2.22.0    2024-10-29 [1] Bioconductor 3.20 (R 4.4.3)
##  rstudioapi            0.17.1    2024-10-22 [1] CRAN (R 4.4.0)
##  rtracklayer         * 1.66.0    2024-10-29 [1] Bioconductor 3.20 (R 4.4.3)
##  S4Arrays              1.6.0     2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  S4Vectors           * 0.44.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  scales                1.3.0     2023-11-28 [2] CRAN (R 4.4.3)
##  sessioninfo           1.2.3     2025-02-05 [2] CRAN (R 4.4.3)
##  SparseArray           1.6.2     2025-02-20 [2] Bioconductor 3.20 (R 4.4.3)
##  stringi               1.8.7     2025-03-27 [1] CRAN (R 4.4.0)
##  stringr             * 1.5.1     2023-11-14 [2] CRAN (R 4.4.3)
##  SummarizedExperiment  1.36.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  tibble              * 3.2.1     2023-03-20 [2] CRAN (R 4.4.3)
##  tidyr               * 1.3.1     2024-01-24 [2] CRAN (R 4.4.3)
##  tidyselect            1.2.1     2024-03-11 [2] CRAN (R 4.4.3)
##  tidyverse           * 2.0.0     2023-02-22 [1] CRAN (R 4.4.0)
##  timechange            0.3.0     2024-01-18 [2] CRAN (R 4.4.3)
##  tinytex               0.56      2025-02-26 [2] CRAN (R 4.4.3)
##  tzdb                  0.4.0     2023-05-12 [2] CRAN (R 4.4.3)
##  UCSC.utils            1.2.0     2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  utf8                  1.2.6     2025-06-08 [1] CRAN (R 4.4.0)
##  vctrs                 0.6.5     2023-12-01 [2] CRAN (R 4.4.3)
##  vroom                 1.6.5     2023-12-05 [2] CRAN (R 4.4.3)
##  withr                 3.0.2     2024-10-28 [1] CRAN (R 4.4.0)
##  WriteXLS              6.8.0     2025-05-22 [1] CRAN (R 4.4.0)
##  xfun                  0.53      2025-08-19 [1] CRAN (R 4.4.0)
##  XML                   3.99-0.19 2025-08-22 [1] CRAN (R 4.4.0)
```

```
##  XVector               0.46.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##  yaml                  2.3.10    2024-07-26 [1] CRAN (R 4.4.0)
##  zlibbioc              1.52.0    2024-10-29 [2] Bioconductor 3.20 (R 4.4.3)
##
##  [1] /usr4/bf527/mgdouq/R/x86_64-pc-linux-gnu-library/4.4
##  [2] /share/pkg.8/r/4.4.3/install/lib64/R/library
##  * -- Packages attached to the search path.
##
## ------------------------------------------------------------------------------
```