Analysis_reproduced

Chandrima

2025-06-16

Load library

```
# Loading the packages required for preprocessing
suppressPackageStartupMessages({
  library(ggplot2)
  library(ggrepel)
  library(RColorBrewer)
  library(DESeq2)
  library(openxlsx)
  library(cowplot)
  library(readr)
  library(edgeR)
  library(rhdf5)
  library(stringr)
  library(sva)
  library(vsn)
  library(genefilter)
  library(TxDb.Mmusculus.UCSC.mm10.knownGene)
  library(rGREAT)
  library(org.Mm.eg.db)
  library(ChIPpeakAnno)
  library(tidyverse)
  library(pheatmap)})
```

```
## Warning: package 'GenomicFeatures' was built under R version 4.3.3
```

Sourcing Functions

```
source("C:/practice_bulk/pre_processing_data.R")
source("C:/practice_bulk/Quality_control.R")
source("C:/practice_bulk/Run_dge.R" )
source("C:/practice_bulk/gene_annotation.R")
source("C:/practice_bulk/Differential_Peaks.R")
```

Sourcing data from Deciphering the role of histone modifications in memory and exhausted CD8 T cells https://www.nature.com/articles/s41598-025-99804-0

```
path <- "C:/practice_bulk/"
dfCounts.H3K4me <- read_tsv(paste0(path, "GSE285245_RawRC_H3K4me3_cleanID.txt"))
```

Loading CUT&RUN data first from the paper

```
## Rows: 50322 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (3): Geneid, Chr, Strand
## dbl (19): Start, End, Length, Arm_P14_Day30_H3K4me3_Rep1_S5, Arm_P14_Day30_H...
##
## 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.
```

```
dfCounts.H3K9me3 <- read_tsv(paste0(path, "GSE285245_RawRC_H3K9me3_cleanID.txt"))
```

```
## Rows: 25434 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (3): Geneid, Chr, Strand
## dbl (19): Start, End, Length, Arm_P14_Day30_H3K9me3_Rep1_S3, Arm_P14_Day30_H...
##
## 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.
```

```
dfCounts.H3K27ac <- read_tsv(paste0(path, "GSE285245_RawRC_H3K27ac_cleanID.txt"))
```

```
## Rows: 38662 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (3): Geneid, Chr, Strand
## dbl (19): Start, End, Length, Arm_P14_Day30_H3K27ac_Rep1_S5, Arm_P14_Day30_H...
##
## 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.
```

```
dfCounts.H3K27me3 <- read_tsv(paste0(path, "GSE285245_RawRC_H3K27me3_cleanID.txt"))
```

```
## Rows: 32659 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (3): Geneid, Chr, Strand
## dbl (19): Start, End, Length, Arm_P14_Day30_H3K27me3_Rep1_S7, Arm_P14_Day30_...
##
## 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.
```

```
# Bedfiles
bedData.H3K4me <- dfCounts.H3K4me[,2:6]
bedData.H3K9me3 <- dfCounts.H3K9me3[,2:6]
bedData.H3K27ac <- dfCounts.H3K27ac[,2:6]
bedData.H3K27me3 <- dfCounts.H3K27me3[,2:6]
```

```
#make the rownames as chromosome loci
dfCounts.H3K4me <- SetRow(dfCounts.H3K4me)
dfConditions.H3K4me <- SetCondition(dfCounts.H3K4me)

dfCounts.H3K9me3 <- SetRow(dfCounts.H3K9me3)
dfConditions.H3K9me3 <- SetCondition(dfCounts.H3K9me3)

dfCounts.H3K27ac <- SetRow(dfCounts.H3K27ac)
dfConditions.H3K27ac <- SetCondition(dfCounts.H3K27ac)

dfCounts.H3K27me3 <- SetRow(dfCounts.H3K27me3)
dfConditions.H3K27me3 <- SetCondition(dfCounts.H3K27ac)
```

Running basic pre-processing of the counts matrix In the paper no additional filtering was done for CUT&RUN sequencing data

So we direct run the deseq

```
deRNA.H3K4me <- RunDeseq(dfCounts.H3K4me, dfConditions.H3K4me)
```

```
## converting counts to integer mode

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
deRNA.H3K9me3 <- RunDeseq(dfCounts.H3K9me3, dfConditions.H3K9me3)
```

```
## converting counts to integer mode

## estimating size factors

## estimating dispersions
```

```
## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
deRNA.H3K27ac <- RunDeseq(dfCounts.H3K27ac,dfConditions.H3K27ac)
```

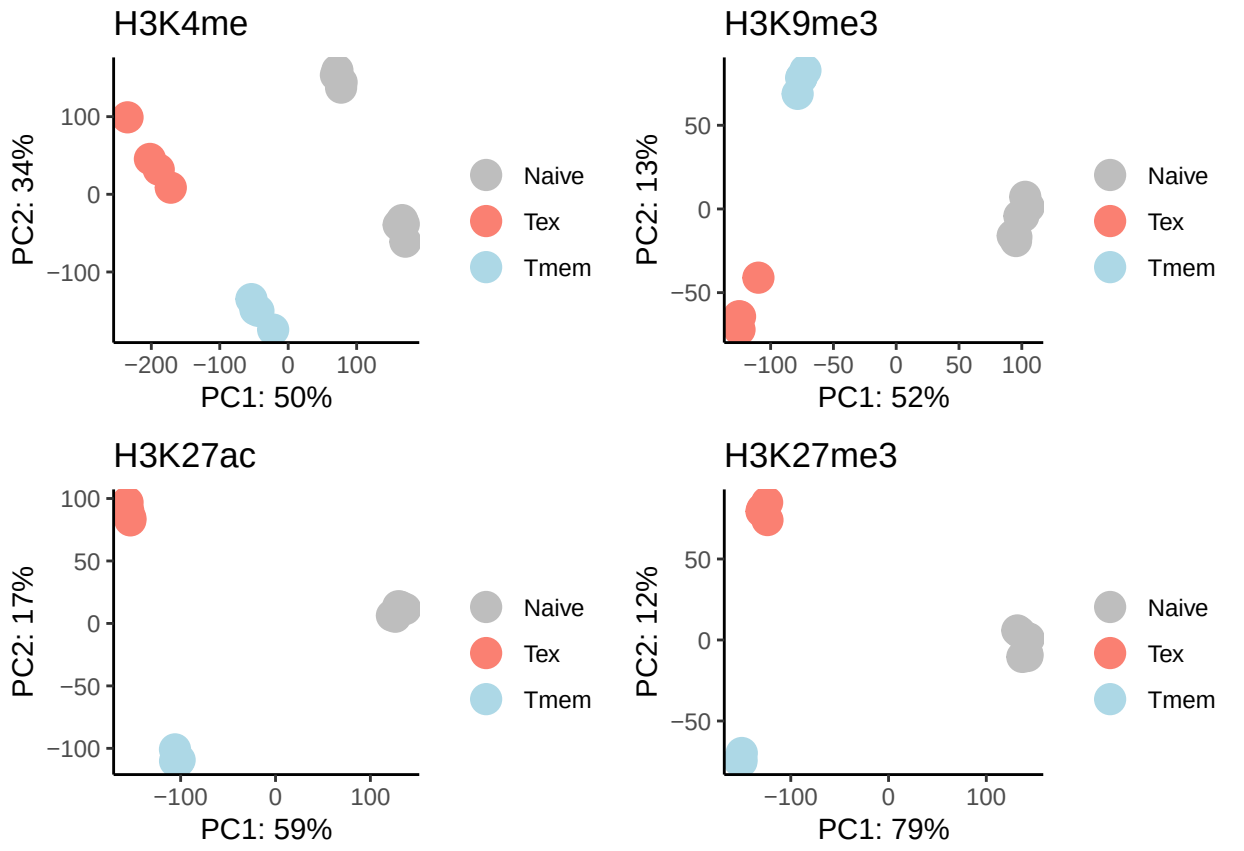
```
## converting counts to integer mode

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
deRNA.H3K27me3 <- RunDeseq(dfCounts.H3K27me3 , dfConditions.H3K27me3)
```

```
## converting counts to integer mode

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

Now we run PCA on the DESeq2 Normalized data

```r
# Normalized data
dfNormalizedCounts.H3K4me <- as.data.frame(counts(deRNA.H3K4me, normalized = T))
p.H3K4me <- RunPCA(dfNormalizedCounts.H3K4me, dfConditions.H3K4me)+ggtitle("H3K4me")

dfNormalizedCounts.H3K9me3<- as.data.frame(counts(deRNA.H3K9me3, normalized = T))
p.H3K9me3 <- RunPCA(dfNormalizedCounts.H3K9me3, dfConditions.H3K9me3)+ggtitle("H3K9me3")

dfNormalizedCounts.H3K27ac <- as.data.frame(counts(deRNA.H3K27ac, normalized = T))
p.H3K27ac <- RunPCA(dfNormalizedCounts.H3K27ac, dfConditions.H3K27ac)+ggtitle("H3K27ac")

dfNormalizedCounts.H3K27me3 <- as.data.frame(counts(deRNA.H3K27me3, normalized = T))
p.H3K27me3 <- RunPCA(dfNormalizedCounts.H3K27me3, dfConditions.H3K27me3)+ggtitle("H3K27me3")

plot_grid(ncol = 2, p.H3K4me,p.H3K9me3,p.H3K27ac, p.H3K27me3)
```



As is visible the analysis was reproduced to the papers figure 1 panel b. Now we compute differentially expressed peak regions for all the sample, we started with making pairwise comparison dataframe which will used to loop for the combinations

```r
# Taking in combinations of two samples to create pairwise samples
dfConditions.H3K27ac$celltyp <- as.factor(dfConditions.H3K27ac$celltyp)
dfPairwiseCond <- as.data.frame(combn(levels(as.factor(dfConditions.H3K27ac$celltyp)), 2))
# Making labels for plots
groupLabels <- sapply(dfPairwiseCond, function(x) {
  paste0(x[[1]], "vs", x[[2]])})
```

One the sample was not runned because batch correction issue

```
DPE.H3K9me3 <- RunDGE(deRNA.H3K9me3, dfConditions.H3K9me3)
```

```
## [1] "NaivevsTex"
## [1] "NaivevsTmem"
## [1] "TexvsTmem"
```

```
DPE.H3K27ac <- RunDGE(deRNA.H3K27ac, dfConditions.H3K27ac)
```

```
## [1] "NaivevsTex"
## [1] "NaivevsTmem"
## [1] "TexvsTmem"
```

```
DPE.H3K27me3 <- RunDGE(deRNA.H3K27me3, dfConditions.H3K27me3)
```

```
## [1] "NaivevsTex"
## [1] "NaivevsTmem"
## [1] "TexvsTmem"
```

Now we RNA seq Data pre processing and DGE Loading the counts matrix data

```
dfCounts <- read_tsv(paste0(path, "GSE285248_RawRC_RNA.txt"))
```

```
## Rows: 55471 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (4): Chr, Geneid, Gene.Name, Strand
## dbl (18): Start, End, Naive_P14_Day30_RNA_Rep1_S1, Naive_P14_Day30_RNA_Rep2_...
##
## 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.
```

```
# Saving the first few columns as bed files
bedData <- dfCounts[,1:6]

# Removing any duplicate gene names  before setting them as rownames
dfCounts <- dfCounts[!duplicated(dfCounts$Gene.Name),]

# Making the counts matrix
dfCounts <- dfCounts |>
  select(c(5,7:ncol(dfCounts))) |>
  column_to_rownames("Gene.Name")
```

Making the experiment design and removing low quality reads setting the cut more than 5 as per the paper

```
dfConditions <- SetCondition(dfCounts)
# Setting the design as factor
dfConditions$celltyp <- as.factor(dfConditions$celltyp)
dfConditions$time <- as.factor(dfConditions$time)

# Dataframe, margin(1 for row, 2 for column) in the apply function
dfCounts <- subset(dfCounts, subset = apply(dfCounts, 1,max) > 5)
```

Running DESeq

```
deRNA <- RunDeseq(dfCounts, dfConditions)
```

```
## converting counts to integer mode

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
DGE.rna <- RunDGE(deRNA, dfConditions)
```

Now we run DGE

```
## [1] "NaivevsTex"
## [1] "NaivevsTmem"
## [1] "TexvsTmem"
```

Now we will make the plot in Figure 1 panel c

We start with making the data frame with cutoffs of p_adj <0.05 and absolute logFC > 1.5 for memory vs exhausted rna expression

```
# Select Padj and FC columns of Tex vs Tmem comparison
dge_sig.rna <- as.data.frame(cbind(DGE.rna$qvals$TexvsTmem_padj,DGE.rna$fc$TexvsTmem_fc))
# Setting column names
colnames(dge_sig.rna) <- c("RNA_padj", "RNA_FC")
#Setting rownames
rownames(dge_sig.rna) <- rownames(DGE.rna$qvals)
# Subetting the DGE based upon the cut-off
dge_sig.rna <- subset(dge_sig.rna, RNA_padj < 0.05 & abs(RNA_FC) > 1.5)

# Saving the rownames as column gene
dge_sig.rna <- dge_sig.rna |> rownames_to_column("gene")
```

Now we do the same for H3K27ac and H3K27me3

```r
# For H3K27ac
dge_sig.H3K27ac <- as.data.frame(cbind(DPE.H3K27ac$qvals$TexvsTmem_padj,DPE.H3K27ac$fc$TexvsTmem_fc))
colnames(dge_sig.H3K27ac) <- c("H3K27ac_padj", "H3K27ac_FC")
rownames(dge_sig.H3K27ac) <- rownames(DPE.H3K27ac$qvals)
dge_sig.H3K27ac <- subset(dge_sig.H3K27ac, H3K27ac_padj < 0.05 & abs(H3K27ac_FC) > 1.5)
dge_sig.H3K27ac <- dge_sig.H3K27ac |> rownames_to_column("ID")
#H3K27me3
dge_sig.H3K27me3 <- as.data.frame(cbind(DPE.H3K27me3$qvals$TexvsTmem_padj,DPE.H3K27me3$fc$TexvsTmem_fc))
colnames(dge_sig.H3K27me3) <- c("H3K27me3_padj", "H3K27me3_FC")
rownames(dge_sig.H3K27me3) <- rownames(DPE.H3K27me3$qvals)
dge_sig.H3K27me3 <- subset(dge_sig.H3K27me3, H3K27me3_padj < 0.05 & abs(H3K27me3_FC) > 1.5)
dge_sig.H3K27me3 <- dge_sig.H3K27me3 |> rownames_to_column("ID")
```

Now we make bed data frame and get the gene annotation

```r
# For H3K27ac
H3K27ac.associated_genes <- GeneAnnot(dge_sig.H3K27ac)
```

```
## * TSS source: TxDb.

## * check whether TxDb package 'TxDb.Mmusculus.UCSC.mm10.knownGene' is installed.

## * gene ID type in the extended TSS is 'Entrez Gene ID'.

## * restrict chromosomes to 'chr1, chr2, chr3, chr4, chr5, chr6, chr7, chr8, chr9, chr10,
##     chr11, chr12, chr13, chr14, chr15, chr16, chr17, chr18, chr19, chrX, chrY, chrM'.

## * 20585/24515 protein-coding genes left.

## * update seqinfo to the selected chromosomes.

## * TSS extension mode is 'basalPlusExt'.

## * construct the basal domains by extending 5000bp to upstream and 1000bp to downsteram of TSS.

## * calculate distances to neighbour regions.

## * extend to both sides until reaching the neighbour genes or to the maximal extension.

## duplicated or NA names found. Rename all the names by numbers.

## * use GO:BP ontology with 15848 gene sets (source: org.Mm.eg.db).

## * check gene ID type in 'gene_sets' and in 'extended_tss'.

## * use whole genome as background.

## * remove excluded regions from background.
```

```
## * overlap ‘gr‘ to background regions (based on midpoint).

## * in total 3146 ‘gr‘.

## * overlap extended TSS to background regions.

## * check which genes are in the gene sets.

## * only take gene sets with size >= 5.

## * in total 9464 gene sets.

## * overlap ‘gr‘ to every extended TSS.

## * perform binomial test for each biological term.
```

```
# Inner join the data frames of Fold Change and annotated genes to the peak regions
dge_sig.H3K27ac <- inner_join(dge_sig.H3K27ac, H3K27ac.associated_genes, by = "ID")

# We split the multiple genes with each peak region and set the colname as gene
dge_sig.H3K27ac <- unnest(dge_sig.H3K27ac, annotated_genes)
colnames(dge_sig.H3K27ac)[4] <- "gene"

# For H3K27me3
H3K27me3.associated_genes <- GeneAnnot(dge_sig.H3K27me3)
```

```
## * TSS source: TxDb.

## * extended_tss is already cached, directly use it.

## duplicated or NA names found. Rename all the names by numbers.

## * use GO:BP ontology with 15848 gene sets (source: org.Mm.eg.db).

## * check gene ID type in ‘gene_sets‘ and in ‘extended_tss‘.

## * use whole genome as background.

## * remove excluded regions from background.

## * overlap ‘gr‘ to background regions (based on midpoint).

## * in total 1347 ‘gr‘.

## * overlap extended TSS to background regions.

## * check which genes are in the gene sets.

## * only take gene sets with size >= 5.
```

```
## * in total 9464 gene sets.

## * overlap 'gr' to every extended TSS.

## * perform binomial test for each biological term.
```

```
# Inner join the data frames of Fold Change and annotated genes to the peak regions
dge_sig.H3K27me3 <- inner_join(dge_sig.H3K27me3, H3K27me3.associated_genes, by = "ID")

# We split the multiple genes with each peak region and set the colname as gene
dge_sig.H3K27me3 <- unnest(dge_sig.H3K27me3, annotated_genes)
colnames(dge_sig.H3K27me3)[4] <- "gene"
```

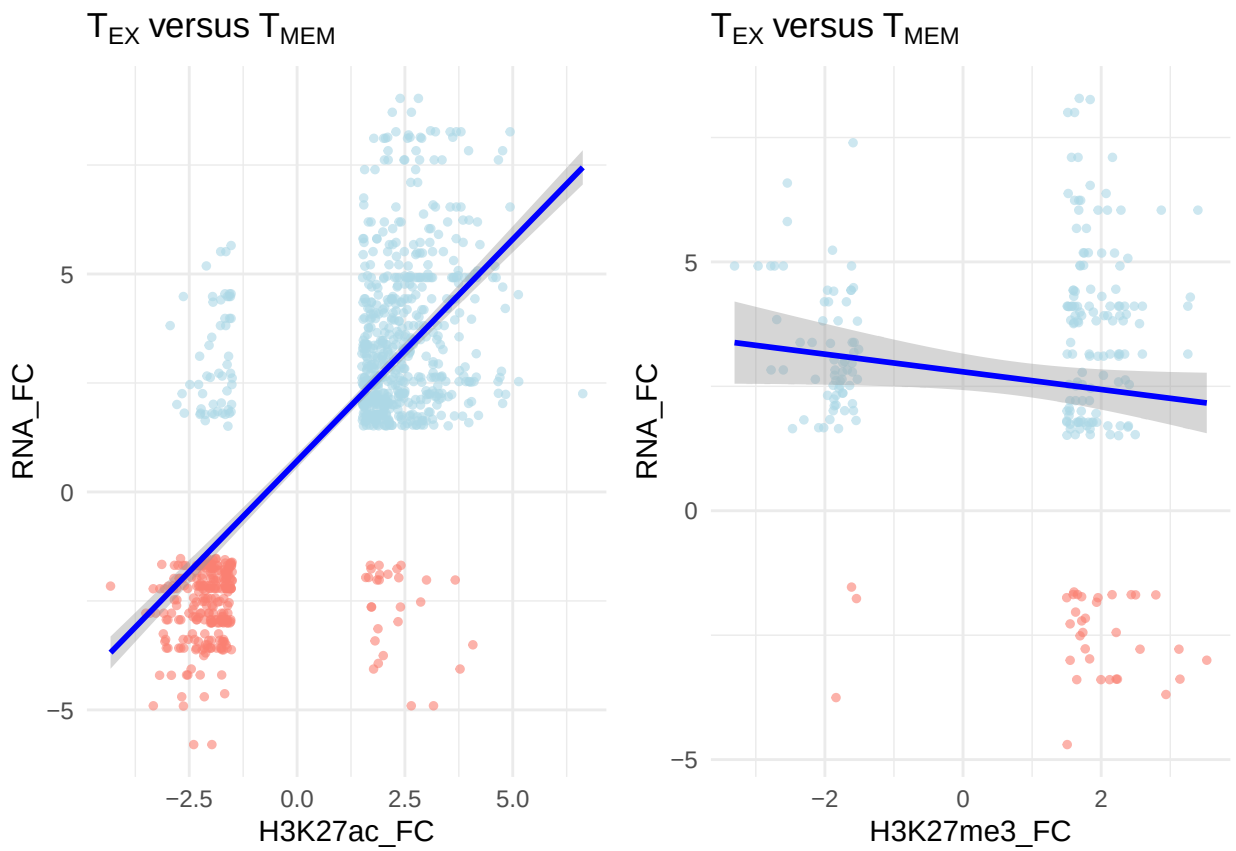
Now join the the data to RNA DGE to make the plots

```
joined_df.rna.H3K27ac <- inner_join(dge_sig.H3K27ac, dge_sig.rna, by = "gene")
joined_df.rna.H3K27me3 <- inner_join(dge_sig.H3K27me3, dge_sig.rna, by = "gene")
```

#Plots

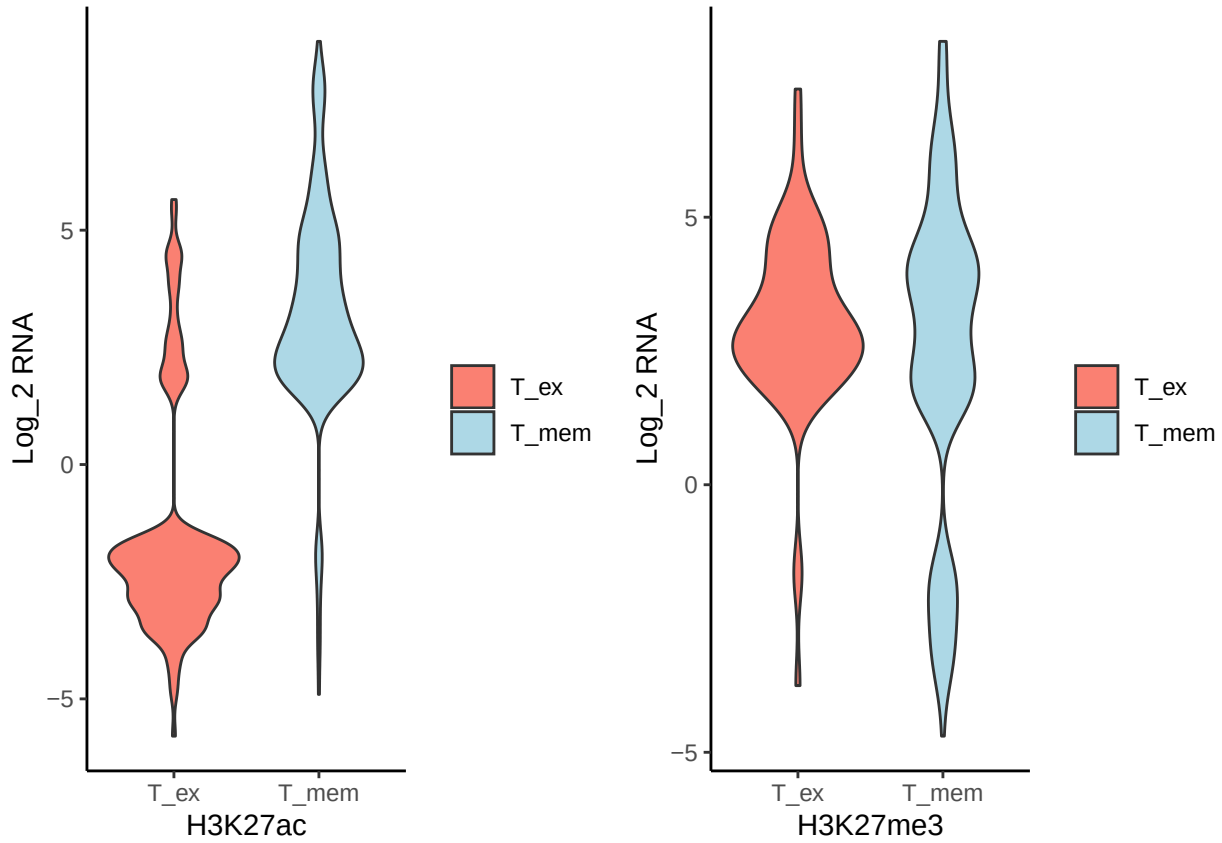
```
# H3K27ac
joined_df.rna.H3K27ac <- ColorScheme(joined_df.rna.H3K27ac)
H3K27ac.corplot <- plotCorrelation(joined_df.rna.H3K27ac, "H3K27ac_FC", "RNA_FC", "color")

# H3K27me3
joined_df.rna.H3K27me3 <- ColorScheme(joined_df.rna.H3K27me3)
H3K27me3.corplot <- plotCorrelation(joined_df.rna.H3K27me3, "H3K27me3_FC", "RNA_FC", "color")

plot_grid(ncol = 2, H3K27ac.corplot,H3K27me3.corplot)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



From the figure 1 panel c only one them was reproducible.

```
# Setting the groups based upon the folchange sign for logfold change in HPTMs
joined_df.rna.H3K27ac <- joined_df.rna.H3K27ac |> mutate(grp = if_else(H3K27ac_FC <0 , "T_ex", "T_mem"))
joined_df.rna.H3K27me3 <- joined_df.rna.H3K27me3 |> mutate(grp = if_else(H3K27me3_FC <0 , "T_ex", "T_mem

#Plots
vplot.H3K27ac <-  ggplot(joined_df.rna.H3K27ac, aes(factor(grp), RNA_FC, fill = grp)) +
  geom_violin() +
  labs(
      x = "H3K27ac",
      y = "Log_2 RNA",
    ) +
  scale_fill_manual(name = "" , values = c("salmon", "lightblue"))+
  theme_classic()

vplot.H3K27me3 <-  ggplot(joined_df.rna.H3K27me3, aes(factor(grp), RNA_FC, fill = grp)) +
  geom_violin() +
  labs(
      x = "H3K27me3",
      y = "Log_2 RNA",
    ) +
  scale_fill_manual(name = "" , values = c("salmon", "lightblue"))+
  theme_classic()
plot_grid(ncol = 2, vplot.H3K27ac, vplot.H3K27me3)
```

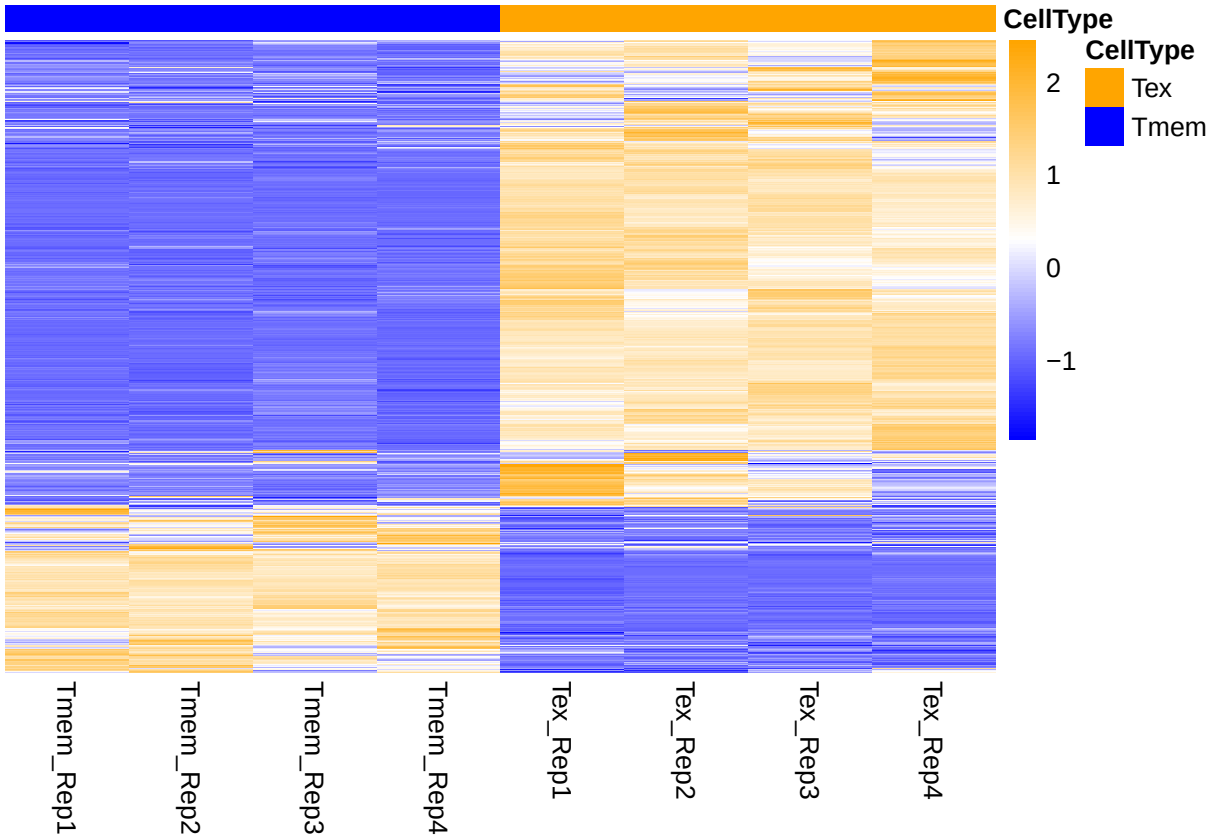


From this plot it can be see for one of the hPTMS plots from the fig 5 panel j was reproducible

Next we reproduce fig 5 panel l heatmap

```
# Step 1: Normalize counts and filter for DE genes
dfNormalizedCounts <- as.data.frame(counts(deRNA, normalized = TRUE))
dfNormalizedCounts <- dfNormalizedCounts[rownames(dfNormalizedCounts) %in% dge_sig.rna$gene, 9:16]

# Step 2: Extract metadata for annotation
setcol <- SetCondition(dfNormalizedCounts)
data <- str_split_fixed(setcol$Samples, "_", 6)
setcol$rep <- data[, 5]
setcol$Group <- paste0(setcol$celltyp, "_", setcol$rep)
colnames(dfNormalizedCounts) <- setcol$Group

# Step 3: Z-score scaling (row-wise)
scale.data <- t(scale(t(dfNormalizedCounts)))

# Step 4: Build annotation_col dataframe
annotation_col <- data.frame(CellType = setcol$celltyp)
rownames(annotation_col) <- colnames(scale.data)

# Step 5: Define annotation colors
ann_colors <- list(CellType = c("Tex" = "orange", "Tmem" = "blue"))
```

```
# Step 6: Plot heatmap with clustered rows and hidden dendrogram
pheatmap(
  scale.data,
  col = colorRampPalette(c("blue", "white", "orange"))(100),
  cluster_rows = TRUE,
  cluster_cols = FALSE,
  treeheight_row = 0,           # Hides dendrogram but keeps clustering
  show_rownames = FALSE,
  annotation_col = annotation_col,
  annotation_colors = ann_colors
)
```



Now look for differential peaks counts accross various hPTMS for Tmem vs Tex

```
#,DPE.H3K27me3$quals$TexvsTmem_padj, DPE.H3K9me3$quals$TexvsTmem_padj
H3K27ac.Sig <- CountDPE(DPE.H3K27ac,"TexvsTmem_H3K27ac")
```

```
## `summarise()` has grouped output by 'comparison'. You can override using the
## `.groups` argument.
```

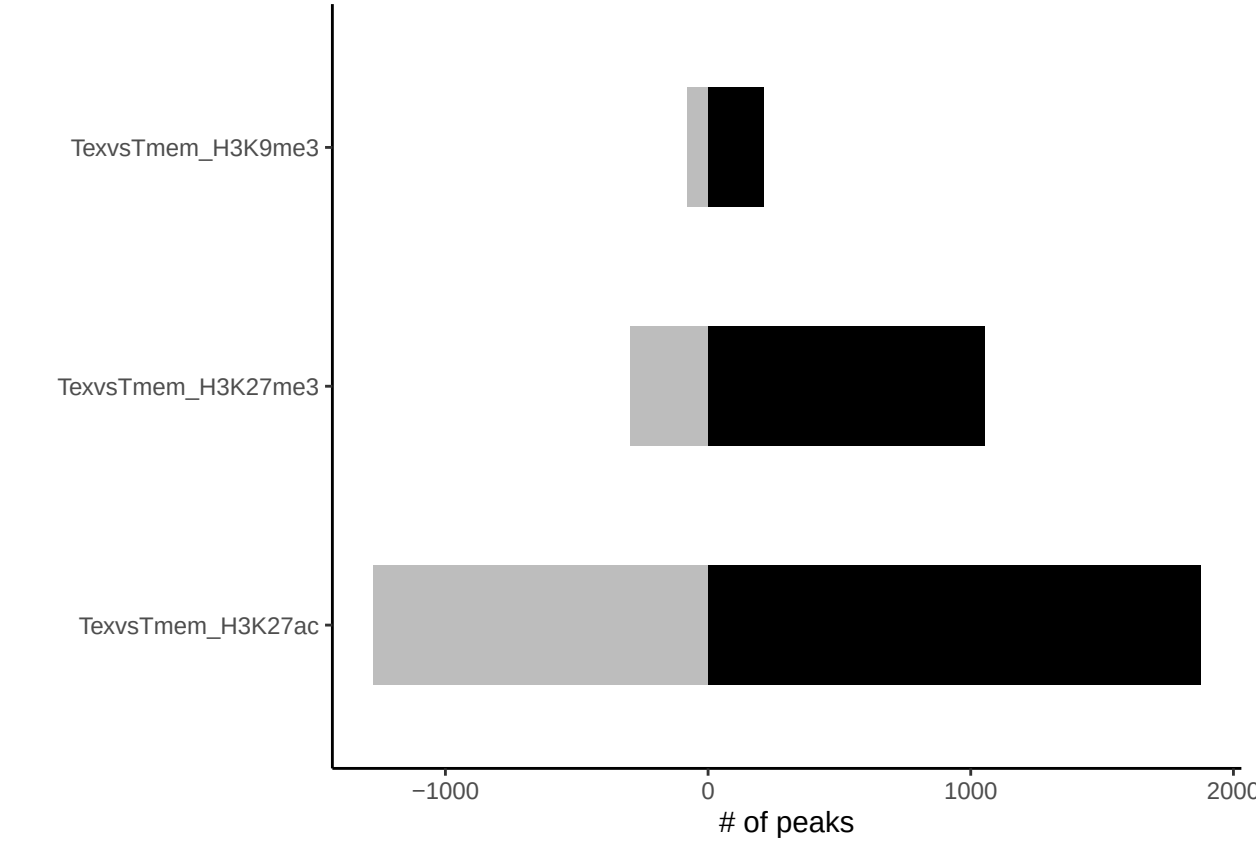
```
H3K27me3.Sig <- CountDPE(DPE.H3K27me3,"TexvsTmem_H3K27me3")
```

```
## `summarise()` has grouped output by 'comparison'. You can override using the
## `.groups` argument.
```

```
H3K9me3.Sig <- CountDPE(DPE.H3K9me3,"TexvsTmem_H3K9me3")
```

```
## `summarise()` has grouped output by 'comparison'. You can override using the
## `.groups` argument.
```

```
all_comparison <- rbind(H3K9me3.Sig,H3K27me3.Sig,H3K27ac.Sig )

ggplot(data = all_comparison, aes(x=com_peaks,y=comparison)) +
  geom_col(aes(fill=com_peaks > 0),width=0.5,show.legend = FALSE) +
  scale_fill_manual(values = c("#BDBDBD","black")) +
  labs(x="# of peaks",y="")+
  theme_classic()
```




```
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 26100)
##
## Matrix products: default
##
```

```
##
## locale:
## [1] LC_COLLATE=English_British Indian Ocean Territory.utf8
## [2] LC_CTYPE=English_British Indian Ocean Territory.utf8
## [3] LC_MONETARY=English_British Indian Ocean Territory.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_British Indian Ocean Territory.utf8
##
## time zone: America/Indianapolis
## tzcode source: internal
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] pheatmap_1.0.12
##  [2] lubridate_1.9.4
##  [3] forcats_1.0.0
##  [4] dplyr_1.1.4
##  [5] purrr_1.0.2
##  [6] tidyr_1.3.1
##  [7] tibble_3.2.1
##  [8] tidyverse_2.0.0
##  [9] ChIPpeakAnno_3.36.1
## [10] org.Mm.eg.db_3.18.0
## [11] rGREAT_2.4.0
## [12] TxDb.Mmusculus.UCSC.mm10.knownGene_3.10.0
## [13] GenomicFeatures_1.54.4
## [14] AnnotationDbi_1.64.1
## [15] vsn_3.70.0
## [16] sva_3.50.0
## [17] BiocParallel_1.36.0
## [18] genefilter_1.84.0
## [19] mgcv_1.9-0
## [20] nlme_3.1-163
## [21] stringr_1.5.1
## [22] rhdf5_2.46.1
## [23] edgeR_4.0.16
## [24] limma_3.58.1
## [25] readr_2.1.5
## [26] cowplot_1.1.3
## [27] openxlsx_4.2.8
## [28] DESeq2_1.42.0
## [29] SummarizedExperiment_1.32.0
## [30] Biobase_2.62.0
## [31] MatrixGenerics_1.14.0
## [32] matrixStats_1.2.0
## [33] GenomicRanges_1.54.1
## [34] GenomeInfoDb_1.38.5
## [35] IRanges_2.36.0
## [36] S4Vectors_0.40.2
## [37] BiocGenerics_0.48.1
## [38] RColorBrewer_1.1-3
```

```
## [39] ggrepel_0.9.5
## [40] ggplot2_3.5.1
##
## loaded via a namespace (and not attached):
##   [1] splines_4.3.2
##   [2] later_1.4.1
##   [3] BiocIO_1.12.0
##   [4] bitops_1.0-7
##   [5] filelock_1.0.3
##   [6] preprocessCore_1.64.0
##   [7] graph_1.80.0
##   [8] XML_3.99-0.18
##   [9] lifecycle_1.0.4
##  [10] doParallel_1.0.17
##  [11] vroom_1.6.5
##  [12] lattice_0.21-9
##  [13] MASS_7.3-60
##  [14] magrittr_2.0.3
##  [15] rmarkdown_2.29
##  [16] yaml_2.3.8
##  [17] httpuv_1.6.15
##  [18] zip_2.3.2
##  [19] DBI_1.2.3
##  [20] abind_1.4-5
##  [21] zlibbioc_1.48.0
##  [22] RCurl_1.98-1.14
##  [23] rappdirs_0.3.3
##  [24] circlize_0.4.16
##  [25] GenomeInfoDbData_1.2.11
##  [26] annotate_1.80.0
##  [27] codetools_0.2-19
##  [28] DelayedArray_0.28.0
##  [29] DT_0.33
##  [30] xml2_1.3.6
##  [31] tidyselect_1.2.1
##  [32] shape_1.4.6.1
##  [33] futile.logger_1.4.3
##  [34] farver_2.1.1
##  [35] BiocFileCache_2.10.2
##  [36] GenomicAlignments_1.38.2
##  [37] GetoptLong_1.0.5
##  [38] multtest_2.58.0
##  [39] survival_3.5-7
##  [40] iterators_1.0.14
##  [41] foreach_1.5.2
##  [42] tools_4.3.2
##  [43] progress_1.2.3
##  [44] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
##  [45] Rcpp_1.0.12
##  [46] glue_1.7.0
##  [47] SparseArray_1.2.3
##  [48] xfun_0.41
##  [49] withr_3.0.2
##  [50] formatR_1.14
```

```
##  [51] BiocManager_1.30.22
##  [52] fastmap_1.2.0
##  [53] rhdf5filters_1.14.1
##  [54] fansi_1.0.6
##  [55] digest_0.6.34
##  [56] timechange_0.3.0
##  [57] R6_2.5.1
##  [58] mime_0.12
##  [59] colorspace_2.1-0
##  [60] GO.db_3.18.0
##  [61] biomaRt_2.58.2
##  [62] RSQLite_2.3.9
##  [63] utf8_1.2.4
##  [64] generics_0.1.3
##  [65] rtracklayer_1.62.0
##  [66] prettyunits_1.2.0
##  [67] InteractionSet_1.30.0
##  [68] httr_1.4.7
##  [69] htmlwidgets_1.6.4
##  [70] S4Arrays_1.2.0
##  [71] regioneR_1.34.0
##  [72] pkgconfig_2.0.3
##  [73] gtable_0.3.4
##  [74] blob_1.2.4
##  [75] XVector_0.42.0
##  [76] htmltools_0.5.7
##  [77] RBGL_1.78.0
##  [78] scales_1.3.0
##  [79] TxDb.Hsapiens.UCSC.hg38.knownGene_3.18.0
##  [80] png_0.1-8
##  [81] knitr_1.45
##  [82] lambda.r_1.2.4
##  [83] rstudioapi_0.17.1
##  [84] tzdb_0.4.0
##  [85] rjson_0.2.23
##  [86] curl_6.1.0
##  [87] org.Hs.eg.db_3.18.0
##  [88] cachem_1.1.0
##  [89] GlobalOptions_0.1.2
##  [90] parallel_4.3.2
##  [91] restfulr_0.0.15
##  [92] pillar_1.9.0
##  [93] grid_4.3.2
##  [94] vctrs_0.6.5
##  [95] promises_1.3.2
##  [96] dbplyr_2.5.0
##  [97] xtable_1.8-4
##  [98] evaluate_1.0.3
##  [99] VennDiagram_1.7.3
## [100] cli_3.6.2
## [101] locfit_1.5-9.8
## [102] compiler_4.3.2
## [103] futile.options_1.0.1
## [104] Rsamtools_2.18.0
```

```
## [105] rlang_1.1.3
## [106] crayon_1.5.2
## [107] labeling_0.4.3
## [108] affy_1.80.0
## [109] stringi_1.8.3
## [110] munsell_0.5.0
## [111] Biostrings_2.70.3
## [112] Matrix_1.6-5
## [113] BSgenome_1.70.2
## [114] hms_1.1.3
## [115] bit64_4.0.5
## [116] Rhdf5lib_1.24.2
## [117] KEGGREST_1.42.0
## [118] statmod_1.5.0
## [119] shiny_1.10.0
## [120] highr_0.10
## [121] memoise_2.0.1
## [122] affyio_1.72.0
## [123] bit_4.0.5
```