SCEPTRE CHIP-seq

2023-03-20

```
#load required packages. Sometimes need to redownload ondisc
library(biomaRt)
library(plyranges)
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, 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, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
```

```
## Loading required package: IRanges
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: GenomeInfoDb
```

```
##
## Attaching package: 'plyranges'
```

```
## The following object is masked from 'package:IRanges':
##
##     slice

## The following object is masked from 'package:biomaRt':
##
##     select

## The following object is masked from 'package:stats':
##
##     filter
```

```
library(GenomicRanges)
library(genomation)
```

```
## Loading required package: grid

## Warning: replacing previous import 'Biostrings::pattern' by 'grid::pattern'
## when loading 'genomation'
```

```
#devtools::install_github('timothy-barry/ondisc')
library(ondisc)
library(sceptre3)
library(BH)
```

```
#read in papalexi data
LOCAL_SCEPTRE2_DATA_DIR <-.get_config_path("LOCAL_SCEPTRE2_DATA_DIR")
papalexi_dir <- paste0(LOCAL_SCEPTRE2_DATA_DIR, "data/papalexi/eccite_screen/")
# gene info
gene_odm_fp <- paste0(papalexi_dir, "gene/matrix.odm")
gene_metadata_fp <- paste0(papalexi_dir, "gene/metadata_qc.rds")
gene_odm <- read_odm(odm_fp = gene_odm_fp, metadata_fp = gene_metadata_fp)
```

```
#get TSS for each gene
ensembl <- useEnsembl(host = 'https://grch37.ensembl.org',biomart = "genes",
                      dataset = "hsapiens_gene_ensembl")
A = getBM(attributes=c("hgnc_symbol", "chromosome_name", "start_position",
                       "end_position", "strand"),
          filters=c('hgnc_symbol'),
          value = gene_odm |> get_feature_ids(), mart=ensembl) |>
  filter(chromosome_name %in% c(1:22, "X", "Y"))
```

```
#get start and end site depending on whether the strand is postive or negative
TSS_start = rep(NA,nrow(A))
TSS_end = rep(NA,nrow(A))
for(j in c(1:nrow(A))){
  #if strand positive, use [start-500,start]
  if(A$strand[j]==1){
    TSS_end[j] = A$start_position[j]
    TSS_start[j] = A$start_position[j]-500
  }else{
```

```
    #if strand negative use [end,end + 500]
    TSS_start[j] = A$end_position[j]
    TSS_end[j] = A$end_position[j]+500
  }
}
#add to A matrix
A$TSS_start = TSS_start
A$TSS_end = TSS_end
#add chr to chromosome name
A$chromosome_name = paste0("chr",A$chromosome_name)
```

```
#use A to make a promoter granges object
promoters <- GRanges(
  seqnames = A$chromosome_name,
  ranges = IRanges(start = A$TSS_start, end = A$TSS_end),
  TF = A$hgnc_symbol)
```

```
#read in chipseq data as granges object
code_dir = .get_config_path("LOCAL_CODE_DIR")
data.dir = paste0(code_dir,"/sceptre2-manuscript/writeups/papalexi_analysis/")
chipseq.dir = paste0(data.dir,
              'GSM935549_hg19_wgEncodeSydhTfbsK562Irf1Ifng6hStdPk.narrowPeak')
chipseq_data = readNarrowPeak(chipseq.dir, track.line=FALSE, zero.based=TRUE)
#get left join of promoters and chipseq peaks
direct_effects = plyranges::join_overlap_left(promoters,chipseq_data)
#get overlapped genes
overlap_genes = direct_effects$TF[is.na(direct_effects$score)==F]
print(paste0("The number of promoters that overlap with Chipseq peaks is ",
             length(overlap_genes)))
```

```
## [1] "The number of promoters that overlap with Chipseq peaks is 6720"
```

```
#read in sceptre and seurat analysis data
sceptre_path = paste0(data.dir,
                      'sceptre_full_mrna_results_with_effect_size.rds')
seurat_path = paste0(data.dir,
                     'seurat_all_perturbations_results.rds')
#read in sceptre and seurat results
sceptre = readRDS(sceptre_path)
seurat = readRDS(seurat_path)
#adjust pvalues according to BH
sceptre$p_value = p.adjust(sceptre$p_value,method = "BH")
for(j in c(1:length(seurat))){
  seurat[[j]]$p_val = p.adjust(seurat[[j]]$p_val,method = "BH")
}
```

```
#get sceptre and seurat IRF1 results
sceptre_IRF1 = subset(sceptre,grna_group == "IRF1")
seurat_IRF1 = seurat[['IRF1']]
#get results for those genes that are associated with IRF1 based on Chipseq
sceptre_IRF1 = subset(sceptre_IRF1,response_id %in%overlap_genes)
seurat_IRF1 = seurat_IRF1[rownames(seurat_IRF1)%in%overlap_genes,]
```

```r
#get number of rejection in this gene set at FDR level 0.05
alpha = 0.05
sceptre_sensitivity = round(mean(sceptre_IRF1$p_value < alpha),3)
seurat_sensitivity = round(mean(seurat_IRF1$p_val < alpha),3)
wrt = "for genes whose promoter overlaps with Chipseq peaks is"
print(paste0("The sensitivity of SCEPTRE ",wrt," is ",sceptre_sensitivity))
```

```
## [1] "The sensitivity of SCEPTRE for genes whose promoter overlaps with Chipseq peaks is is 0.113"
```

```r
print(paste0("The sensitivity of Seurat ",wrt," is ",seurat_sensitivity))
```

```
## [1] "The sensitivity of Seurat for genes whose promoter overlaps with Chipseq peaks is is 0.182"
```