SCEPTRE CHIP-seq

2023-03-20

```r
#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)
library(varhandle)
```

```
##
## Attaching package: 'varhandle'

## The following object is masked from 'package:S4Vectors':
##
##     unfactor
```

```
library(kableExtra)
library(rjson)
```

```
#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"))
```

```r
#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)
```

```r
#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)
```

```r
#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,
                                              minoverlap = 500)
#get overlapped genes
overlap_genes = direct_effects$TF[is.na(direct_effects$score)==F]
null_genes = direct_effects$TF[is.na(direct_effects$score)==T]
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 1644"
```

```r
#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
```

```r
sceptre$p_value_adj = sceptre$p_value
for(val in unique(sceptre$grna_group)){
  ind = which(sceptre$grna_group == val)
  sceptre$p_value_adj[ind] = p.adjust(sceptre$p_value[ind],method = "BH")
}
for(j in c(1:length(seurat))){
  seurat[[j]]$p_val_adj = p.adjust(seurat[[j]]$p_val,method = "BH")
}
```

```r
#get sceptre and seurat IRF1 results
PRTB = 'IRF1'
sceptre_prtb = subset(sceptre,grna_group == PRTB)
seurat_prtb = seurat[[PRTB]]
```

```r
alpha = 0.05
#get all significant genes from each method
sceptre_sig = unfactor(sceptre_prtb$response_id[sceptre_prtb$p_value_adj < alpha])
seurat_sig = rownames(seurat_prtb)[seurat_prtb$p_val_adj < alpha]
#get true positive sets
sceptre_true = sceptre_sig[sceptre_sig %in% overlap_genes]
seurat_true = seurat_sig[seurat_sig %in% overlap_genes]
#get number of shared and unique true postives
shared_true = sceptre_true[sceptre_true %in% seurat_true]
sceptre_true_unique = sceptre_true[(sceptre_true %in% seurat_true) == F]
seurat_true_unique = seurat_true[(seurat_true %in% sceptre_true) == F]

#get sensitivity
sceptre_sensitivity = length(sceptre_true)/length(overlap_genes)
seurat_sensitivity = length(seurat_true)/length(overlap_genes)

#get false positive sets
sceptre_false = sceptre_sig[sceptre_sig %in% null_genes]
seurat_false = seurat_sig[seurat_sig %in% null_genes]
#get number of shared and unique false postives
shared_false = sceptre_false[sceptre_false %in% seurat_false]
sceptre_false_unique = sceptre_false[(sceptre_false %in% seurat_false) == F]
seurat_false_unique = seurat_false[(seurat_false %in% sceptre_false) == F]


#get specificity
sceptre_specificity = 1-length(sceptre_false)/length(null_genes)
seurat_specificity = 1-length(seurat_false)/length(null_genes)
```

```r
#make table for true positives
true_pos = matrix(NA,1,4)
rownames(true_pos) = "True Positives"
colnames(true_pos) = c("Total Positive Genes","Unique to SCEPTRE",
                       "Unique to Seurat","Shared")
true_pos[1,] = c(length(overlap_genes),length(sceptre_true_unique),
                 length(seurat_true_unique),length(shared_true))

results_table = kable(true_pos,booktabs = TRUE, linesep = "",
    caption = paste0("SCEPTRE Vs Seurat:",PRTB," Perturbation True Positives"))
```

Table 1: SCEPTRE Vs Seurat:IRF1 Perturbation True Positives

	Total Positive Genes	Unique to SCEPTRE	Unique to Seurat	Shared
True Positives	1644	66	44	160

Table 2: SCEPTRE Vs Seurat:IRF1 Perturbation False Positives

	Total Null Genes	Unique to SCEPTRE	Unique to Seurat	Shared
False Positives	11483	433	258	1341

```
kable_styling(results_table,position = "center")
```

```
#make table for true positives
false_pos = matrix(NA,1,4)
rownames(false_pos) = "False Positives"
colnames(false_pos) = c("Total Null Genes","Unique to SCEPTRE",
                        "Unique to Seurat","Shared")
false_pos[1,] = c(length(null_genes),length(sceptre_false_unique),
                  length(seurat_false_unique),length(shared_false))

results_table = kable(false_pos,booktabs = TRUE, linesep = "",
    caption = paste0("SCEPTRE Vs Seurat:",PRTB," Perturbation False Positives"))
kable_styling(results_table,position = "center")
```

```
#make table for true positives
sens_spec = matrix(NA,2,2)
rownames(sens_spec) = c("SCEPTRE","Seurat")
colnames(sens_spec) = c("Sensitivity","Specificity")
sens_spec[1,] = c(sceptre_sensitivity,sceptre_specificity)
sens_spec[2,] = c(seurat_sensitivity,seurat_specificity)

results_table = kable(sens_spec,booktabs = TRUE, linesep = "",
caption = paste0("SCEPTRE Vs Seurat:",PRTB,
                 " Perturbation Sensitivity and Specificity"))
kable_styling(results_table,position = "center")
```

```
TF = direct_effects$TF
scores = direct_effects$signalValue
top_genes = sort(TF[order(scores,decreasing = T)][1:500])
```

Table 3: SCEPTRE Vs Seurat:IRF1 Perturbation Sensitivity and Specificity

	Sensitivity	Specificity
SCEPTRE	0.1374696	0.8455108
Seurat	0.1240876	0.8607507

Table 4: SCEPTRE Vs Seurat:IRF1 Perturbation CHIPseq and Reference Agreement

	Unique to CHIPseq	Unique to Reference	Shared
IRF1 Targets	491	493	9

```r
data.dir = paste0(code_dir,"/sceptre2-manuscript/writeups/papalexi_analysis/")
targets.dir = paste0(data.dir,'IRF1_targets.json')
prtb_targets = fromJSON(file = targets.dir)
targets = c()
for(j in c(1:length(prtb_targets$associations))){
  targets = c(targets,prtb_targets$associations[[j]]$gene$symbol)
}
```

```r
overlap_targets = top_genes[(top_genes%in% targets)]
CHIP_target_unique = top_genes[(top_genes%in% targets)==F]
ref_target_unique = targets[(targets %in% top_genes) == F]
#make table for true positives
chip_agree = matrix(NA,1,3)
rownames(chip_agree) = c("IRF1 Targets")
colnames(chip_agree ) = c("Unique to CHIPseq","Unique to Reference","Shared")
chip_agree [1,] = c(length(CHIP_target_unique),length(ref_target_unique),length(overlap_targets))


results_table = kable(chip_agree ,booktabs = TRUE, linesep = "",
caption = paste0("SCEPTRE Vs Seurat:",PRTB,
                 " Perturbation CHIPseq and Reference Agreement"))
kable_styling(results_table,position = "center")
```