Hematopoeisis tutorial - ArchR project

Shang-Yang and Xiaosai Yao

5 May 2023

Package

epiregulon 1.0.22

Contents

1 Introduction

Gene regulatory networks model the underlying gene regulation hierarchies that drive gene expression and observed phenotypes. The main function of the epiregulon R package is to infer TF activity in single cells by constructing a gene regulatory network (regulons). This is achieved through integration of scATAC-seq and scRNA-seq data and incorporation of public bulk TF ChIP-seq data. Links between regulatory elements and their target genes are established by computing correlations between chromatin accessibility and gene expressions.

Current prerequisite for running epiregulon is a ArchR project with pre-computed peak and gene matrices. It is also expected that LSI dimensionality reduction and integration with an scRNA-seq dataset has been performed. The scATAC-seq experiment can be either paired or unpaired with the scRNA-seq dataset as long as they were already integrated by ArchR. The final output of epiregulon is a matrix of TF activities where rows are individual TFs and columns are single cell indexes.

Alternatively, users can now supply peak, gene, and dimensional reduction matrices derived from a MultiAssayExperiment object. This is to be compatible with future GPSA multiome workflow. Epiregulon implements a custom algorithm to derive a more stringent set of P2G correlations compared to ArchR.

In this vignette we demonstrate the workflow of epiregulon along with some visualization functionalities using the tutorial datasets from ArchR development team. In this dataset, scRNAseq and scATACseq were unpaired and integrated by the `addGeneIntegrationMatrix` function.

2 Installation

Epiregulon is currently available on R/dev

```
devtools::load_all()
#library(epiregulon)
```

If you would like to install from gitlab,

```
devtools::install_github(repo='xiaosaiyao/epiregulon')
library(epiregulon)
```

3 Data preparation

Please refer to the full ArchR manual for instructions

Before running Epiregulon, the following analyses need to be completed: 1. Obtain a peak matrix on scATAC-seq by using addGroupCoverages > addReproduciblePeakSet > addPeakMatrix. See chapter 10 from ArchR manual 2. RNA-seq integration. a. For unpaired scATAC-seq, use addGeneIntegrationMatrix. See chapter 8 from ArchR manual b. For multiome data, use addGeneExpressionMatrix. See multiome tutorial 3. Perform dimensionality reduction from with either single modalities or joint scRNA-seq and scATAC-seq using addCombinedDims

To verify that all the necessary matrices are present,

```
library(ArchR,quietly = TRUE)
archR_project_path <- "/gstore/project/lineage/sam/heme_GRN/OUTPUT"
proj <- loadArchRProject(path = archR_project_path, showLogo = FALSE)
getAvailableMatrices(proj)
```

4 Quick start

4.1 Retrieve bulk TF ChIP-seq binding sites

First, we retrieve the information of TF binding sites collected from Cistrome and ENCODE ChIP-seq, which are hosted on Genomitory. Currently, human genomes hg19 and hg38 and mouse genome mm10 are available

```
grl <- getTFMotifInfo(genome = "hg19")
head(grl)
```

4.2 Link ATACseq peaks to target genes

Next, we compute peak to gene correlations using the calculateP2G function from ArchR package. The user would need to supply a path to an ArchR project that already contains the peak matrix, gene expression matrix and Latent semantic indexing (LSI) dimensionality reduction. The example project shown here utilizes the tutorial datasets provided by the ArchR development team.

```
# path to ArchR project
p2g <- calculateP2G(ArchR_path = archR_project_path)
head(p2g)
```

4.3 Add TF motif binding to peaks

The next step is to add the TF motif binding information by overlapping the regions of the peak matrix with the bulk chip-seq database loaded in 2. The user can supply either an archR project path and this function will retrieve the peak matrix, or a peakMatrix in the form of a Granges object or RangedSummarizedExperiment.

```
overlap <- addTFMotifInfo(p2g, grl, archR_project_path = archR_project_path)
head(overlap)
```

4.4 Generate regulons

A long format dataframe, representing the inferred regulons, is then generated. The dataframe consists of three columns:

- tf (transcription factor)
- target gene
- peak to gene correlation between tf and target gene

```
regulon <- getRegulon(p2g, overlap, aggregate=FALSE)
head(regulon)
```

Epiregulon outputs two different correlations. The first, termed “corr”, is the correlation between chromatin accessibility of regulatory elements vs expression of target genes calculated by ArchR. The second, termed “weight”, can be generated by the addWeights function, which compute the correlation between gene expressions of TF vs expressions of target genes, shown below. The user is required to supply the clustering or batch labels of the scRNA-seq dataset when running addWeights. “Weight” is the preferred metric for calculating activity.

load scRNA-seq data

```
sce <- readRDS("/gstore/project/lineage/sam/heme_GRN/scRNA-Granja-2019.rds")
```

Trim regulon for demonstration purposes

```
TFs <- c("FOXA1","GATA3","SOX9", "SPI1")
regulon <- regulon[which(regulon$tf %in% TFs),]
nrow(regulon)
```

Prune network

```
# retrieve gene expression and peak matrix from archR project
GeneExpressionMatrix <- getMatrixFromProject(
    ArchRProj = proj,
    useMatrix = "GeneIntegrationMatrix",
    useSeqnames = NULL,
    verbose = TRUE,
    binarize = FALSE,
    threads = 1,
    logFile = "x"
)

rownames(GeneExpressionMatrix) <- rowData(GeneExpressionMatrix)$name

PeakMatrix <- getMatrixFromProject(
    ArchRProj = proj,
    useMatrix = "PeakMatrix",
    useSeqnames = NULL,
    verbose = TRUE,
    binarize = FALSE,
    threads = 1,
    logFile = "x"
)


pruned.regulon <- pruneRegulon(expMatrix = GeneExpressionMatrix,
                               exp_assay = "GeneIntegrationMatrix",
                               peakMatrix = PeakMatrix,
                               peak_assay = "PeakMatrix",
                               regulon = regulon,
                               clusters = GeneExpressionMatrix$predictedGroup,
                               prune_value = "pval",
                               regulon_cutoff = 0.05)
```

Add Weights to regulon

```
regulon.w <- addWeights(regulon = regulon,
                        expMatrix = sce,
                        clusters = sce$BioClassification,
                        block_factor = NULL,
                        method = "corr")
head(regulon.w)
```

4.5 Calculate TF activity

Finally, the activities for a specific TF in each cell are computed by averaging the weighted expressions of target genes linked to the TF weighted.

$$y = \frac{1}{n}\sum_{i=1}^{n} x_i * weight_i$$

where y is the activity of a TF for a cell n is the total number of targets for a TF x_i is the log count expression of target i where i in $\{$1,2,. . . ,n$\}$ $weight_i$ is the weight of TF and target i

```
score.combine <- calculateActivity(expMatrix = sce,
                                   regulon = regulon.w,
                                   mode = "weight",
                                   method = "weightedMean",
                                   exp_assay = "logcounts")
head(score.combine[,1:10])
```

4.6 Differential TF activity test

We can next determine which TFs exhibit differential activities across cell clusters/groups via the findDifferentialActivity function. This function depends on findMarkers function from scran package and allow the same parameters.

```
da_list <- findDifferentialActivity(activity_matrix = score.combine,
                                    groups = sce$BioClassification,
                                    pval.type = "some",
                                    direction = "up",
                                    test.type = "t")
```

getSigGenes compiles the different test results into a single dataframe and enables user to supply their desired cutoffs for significance and variable to order by.

```
markers <- getSigGenes(da_list, fdr_cutoff = 0.05)
head(markers)
```

4.7 Visualizing TF activities

Epiregulon also provides multiple options for visualizing the inferred TF activities.

tSNE or UMAP plots:

```
plotActivityDim(sce = sce,
                activity_matrix = score.combine,
                tf = c("FOXA1","GATA3","SOX9", "SPI1"),
                dimtype = "TSNE",
                combine = TRUE)
```

Violin plots:

```
plotActivityViolin(activity_matrix = score.combine,
                   tf = c("FOXA1","GATA3","SOX9", "SPI1"),
                   clusters = sce$BioClassification)
```

Bubble plot:

```
plotBubble(activity_matrix = score.combine,
           tf = c("FOXA1","GATA3","SOX9", "SPI1"),
           sce$BioClassification,
           bubblesize = "FDR")
```

5 Session Info

```
sessionInfo()
```