

# Hematopoiesis tutorial - ArchR project

*Shang-Yang and Xiaosai Yao*

16 November 2022

## Package

epiregulon 1.0.18

## Contents

1	Introduction . . . . .	2
2	Installation . . . . .	2
3	Data preparation . . . . .	2
4	Quick start . . . . .	4
4.1	Retrieve bulk TF ChIP-seq binding sites . . . . .	4
4.2	Link ATACseq peaks to target genes . . . . .	4
4.3	Add TF motif binding to peaks . . . . .	5
4.4	Generate regulons . . . . .	6
4.5	Calculate TF activity . . . . .	8
4.6	Differential TF activity test . . . . .	9
4.7	Visualizing TF activities . . . . .	10
5	Session Info . . . . .	13

## 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

```
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 `addGroupCovariates` > `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,

Hematopoiesis tutorial - ArchR project

```
#> [4] "TileMatrix"
```

## 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")
#> redirecting from 'GMTY156:hg19_motif_bed_granges@REVISION-5' to 'GMTY156:hg19_motif_bed_granges@83b9ff4cd
head(grl)
#> GRangesList object of length 6:
#> $`5-hmC`
#> GRanges object with 22860 ranges and 0 metadata columns:
#>           seqnames      ranges strand
#>           <Rle>      <IRanges> <Rle>
#>     [1]     chr1  10001-10685    *
#>     [2]     chr1  13362-13694    *
#>     [3]     chr1  29631-29989    *
#>     [4]     chr1  40454-40754    *
#>     [5]     chr1  135395-135871   *
#>     ...
#>     ...     ...     ...
#> [22856]     chrM  15303-15326    *
#> [22857]     chrM  15328-16172    *
#> [22858]     chrM  16174-16183    *
#> [22859]     chrM  16186-16224    *
#> [22860]     chrM  16226-16492    *
#> -----
#> seqinfo: 25 sequences from an unspecified genome; no seqlengths
#>
#> ...
#> <5 more elements>
```

### 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)
#> Setting ArchRLogging = FALSE
#> Using ArchR to compute peak to gene links...
```

## Hematopoiesis tutorial - ArchR project

```
#> 2022-11-16 22:12:13 : Getting Available Matrices, 0 mins elapsed.
#> 2022-11-16 22:12:14 : Filtered Low Prediction Score Cells (684 of 10250, 0.067), 0.007 mins elapsed.
#> 2022-11-16 22:12:14 : Computing KNN, 0.013 mins elapsed.
#> 2022-11-16 22:12:15 : Identifying Non-Overlapping KNN pairs, 0.023 mins elapsed.
#> 2022-11-16 22:12:17 : Identified 492 Groupings!, 0.063 mins elapsed.
#> 2022-11-16 22:12:17 : Getting Group RNA Matrix, 0.064 mins elapsed.
#> 2022-11-16 22:14:07 : Getting Group ATAC Matrix, 1.888 mins elapsed.
#> 2022-11-16 22:15:40 : Normalizing Group Matrices, 3.442 mins elapsed.
#> 2022-11-16 22:15:44 : Finding Peak Gene Pairings, 3.515 mins elapsed.
#> 2022-11-16 22:15:45 : Computing Correlations, 3.525 mins elapsed.
#> 2022-11-16 22:15:51 : Completed Peak2Gene Correlations!, 3.635 mins elapsed.
head(p2g)
#>   idxATAC chr start    end idxRNA      target Correlation distance
#> 1     7 chr1 801002 801502      2 LINC00115  0.8324892  38350
#> 2     24 chr1 894453 894953     6 KLHL17   0.5261049  1264
#> 3     24 chr1 894453 894953    14 TNFRSF18  0.5125436  247386
#> 5     25 chr1 894960 895460     9 ISG15    0.6024232  53637
#> 4     25 chr1 894960 895460    14 TNFRSF18  0.5509001  246879
#> 6     36 chr1 934450 934950    17 B3GALT6  0.5289083  232929
#>          FDR
#> 1 6.106975e-125
#> 2 5.377552e-35
#> 3 5.592283e-33
#> 5 2.843810e-48
#> 4 6.249324e-39
#> 6 2.005290e-35
```

### 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)
#> Successfully loaded ArchRProject!
#> Computing overlap...
#> Success!
head(overlap)
#>   idxATAC idxTF      tf
#> 1018     7    35 ARNT
#> 1019     7    50 ATF2
#> 1020     7    55 ATF7
#> 1021     7    76 BCL6
#> 1022     7    80 BCOR
#> 1023     7    82 BHLHE40
```

## 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)
#>   idxATAC idxTF      tf  chr  start    end idxRNA    target      corr distance
#> 1       7     ARNT chr1 801002 801502      2 LINC00115 0.8324892 38350
#> 2       7     ATF2 chr1 801002 801502      2 LINC00115 0.8324892 38350
#> 3       7     ATF7 chr1 801002 801502      2 LINC00115 0.8324892 38350
#> 4       7     BCL6 chr1 801002 801502      2 LINC00115 0.8324892 38350
#> 5       7     BCOR chr1 801002 801502      2 LINC00115 0.8324892 38350
#> 6       7     BHLHE40 chr1 801002 801502      2 LINC00115 0.8324892 38350
#>
#>   FDR
#> 1 6.106975e-125
#> 2 6.106975e-125
#> 3 6.106975e-125
#> 4 6.106975e-125
#> 5 6.106975e-125
#> 6 6.106975e-125
```

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)
#> [1] 48931
```

Prune network

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

## Hematopoiesis tutorial - ArchR project

```
    threads = 1,
    logFile = "x"
)
#> 2022-11-16 22:17:26 : Organizing colData, 0.693 mins elapsed.
#> 2022-11-16 22:17:26 : Organizing rowData, 0.695 mins elapsed.
#> 2022-11-16 22:17:26 : Organizing rowRanges, 0.695 mins elapsed.
#> 2022-11-16 22:17:26 : Organizing Assays (1 of 1), 0.695 mins elapsed.
#> 2022-11-16 22:17:28 : Constructing SummarizedExperiment, 0.726 mins elapsed.
#> 2022-11-16 22:17:29 : Finished Matrix Creation, 0.743 mins elapsed.

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

PeakMatrix <- getMatrixFromProject(
    ArchRProj = proj,
    useMatrix = "PeakMatrix",
    useSeqnames = NULL,
    verbose = TRUE,
    binarize = FALSE,
    threads = 1,
    logFile = "x"
)
#> 2022-11-16 22:17:42 : Organizing colData, 0.214 mins elapsed.
#> 2022-11-16 22:17:42 : Organizing rowData, 0.215 mins elapsed.
#> 2022-11-16 22:17:42 : Organizing rowRanges, 0.216 mins elapsed.
#> 2022-11-16 22:17:42 : Organizing Assays (1 of 1), 0.216 mins elapsed.
#> 2022-11-16 22:17:42 : Constructing SummarizedExperiment, 0.223 mins elapsed.
#> 2022-11-16 22:17:47 : Finished Matrix Creation, 0.292 mins elapsed.

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)
#> pruning network with binom tests using a regulon cutoff of pval<0.05
#> binarizing matrices
#> pruning regulons
#>
| | 0%
| |
| |
|=====
|===== 25%
| |
|=====
|===== 50%
| |
|=====
|===== 75%
| |
|===== 100%
```

## Hematopoiesis tutorial - ArchR project

Add Weights to regulon

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

#> adding weights using corr
#> calculating average expression across clusters...
#> computing weights...
#>
| | 0%
| |
|-----| 25%
|
|-----| 50%
|
|-----| 75%
|
|-----| 100%
head(regulon.w)
#>      idxATAC idxTF   tf  chr  start    end idxRNA     target      corr distance
#> 10        7  296 FOXA1 chr1 801002 801502       2 LINC00115 0.8324892  38350
#> 235       24  296 FOXA1 chr1 894453 894953       6 KLHL17 0.5261049  1264
#> 236       24  296 FOXA1 chr1 894453 894953      14 TNFRSF18 0.5125436 247386
#> 1302      36  296 FOXA1 chr1 934450 934950      17 B3GALT6 0.5289083 232929
#> 1552      37  296 FOXA1 chr1 935289 935789       8 HES4 0.6280933   13
#> 2459      46  296 FOXA1 chr1 955267 955767       6 KLHL17 0.6712327 59550
#>          FDR      weight
#> 10  6.106975e-125 -0.18563217
#> 235  5.377552e-35 -0.10638676
#> 236  5.592283e-33 -0.07520848
#> 1302 2.005290e-35 -0.33654071
#> 1552 1.329724e-53 -0.04628539
#> 2459 8.582330e-64 -0.10638676
```

## 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  $i \in \{1, 2, \dots, n\}$   $weight_i$  is the weight of TF and target  $i$

```
score.combine <- calculateActivity(expMatrix = sce,
                                     regulon = regulon.w,
                                     mode = "weight",
```

```

method = "weightedMean",
exp_assay = "logcounts")
#> calculating TF activity from regulon using weightedmean
head(score.combine[,1:10])
#>      CD34_32_R5:AAACCTGAGTATCGAA-1 CD34_32_R5:AAACCTGAGTCGTTG-1
#> FOXA1           0.003168022          0.006498154
#> GATA3          -0.009787296          -0.012589268
#> SOX9           -0.001414632          -0.005367695
#> SPI1            -0.002576604          0.001368212
#>      CD34_32_R5:AAACCTGGTTCACAA-1 CD34_32_R5:AAACGGGAGCTTCGCG-1
#> FOXA1           0.0009826984         0.002231260
#> GATA3          -0.0073062285         -0.008734045
#> SOX9           -0.0005736017         -0.002173620
#> SPI1            -0.0023318937         -0.002738478
#>      CD34_32_R5:AAACGGGAGGGAGTAA-1 CD34_32_R5:AAACGGGAGTTACGGG-1
#> FOXA1           0.0011954923         0.0008346354
#> GATA3          -0.0060802331         -0.0057561205
#> SOX9           -0.0003999532         0.0004751306
#> SPI1            -0.0017820531         -0.0028718975
#>      CD34_32_R5:AAACGGGCAAGTAATG-1 CD34_32_R5:AAACGGGCAAGTTCTG-1
#> FOXA1           -0.0004161807        0.0007104586
#> GATA3          -0.0031771039        -0.0056730850
#> SOX9            0.0019708598        0.0002585261
#> SPI1            -0.0020193533        -0.0020023563
#>      CD34_32_R5:AAACGGGCACAGACAG-1 CD34_32_R5:AAACGGGTAACGTTC-1
#> FOXA1           0.001658212          0.001735316
#> GATA3          -0.010771045          -0.008507215
#> SOX9            -0.002863604          -0.003291236
#> SPI1            -0.004105925          -0.001764217

```

## 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)
#> Using a logFC cutoff of 0 for class 01_HSC
#> Using a logFC cutoff of 0 for class 02_Early.Eryth
#> Using a logFC cutoff of 0 for class 03_Late.Eryth
#> Using a logFC cutoff of 0 for class 04_Early.Baso

```

## Hematopoiesis tutorial - ArchR project

```
#> Using a logFC cutoff of 0 for class 05_CMP.LMPP
#> Using a logFC cutoff of 0 for class 06_CLP.1
#> Using a logFC cutoff of 0 for class 07_GMP
#> Using a logFC cutoff of 0 for class 08_GMP.Neut
#> Using a logFC cutoff of 0 for class 09_pDC
#> Using a logFC cutoff of 0 for class 10_cDC
#> Using a logFC cutoff of 0 for class 11_CD14.Mono.1
#> Using a logFC cutoff of 0 for class 12_CD14.Mono.2
#> Using a logFC cutoff of 0 for class 13_CD16.Mono
#> Using a logFC cutoff of 0 for class 14_Unk
#> Using a logFC cutoff of 0 for class 15_CLP.2
#> Using a logFC cutoff of 0 for class 16_Pre.B
#> Using a logFC cutoff of 0 for class 17_B
#> Using a logFC cutoff of 0 for class 18_Plasma
#> Using a logFC cutoff of 0 for class 19_CD8.N
#> Using a logFC cutoff of 0 for class 20_CD4.N1
#> Using a logFC cutoff of 0 for class 21_CD4.N2
#> Using a logFC cutoff of 0 for class 22_CD4.M
#> Using a logFC cutoff of 0 for class 23_CD8.EM
#> Using a logFC cutoff of 0 for class 24_CD8.CM
#> Using a logFC cutoff of 0 for class 25_NK
#> Using a logFC cutoff of 0 for class 26_Unk
head(markers)
#>           p.value      FDR summary.logFC      class      tf
#> 1  3.883871e-03  1.553549e-02  0.0001311715  01_HSC FOXA1
#> 2  4.128807e-146 1.651523e-145  0.0014115613  02_Early.Eryth FOXA1
#> 3  8.450047e-102 3.380019e-101  0.0020925809  03_Late.Eryth FOXA1
#> 4  9.893183e-43  3.957273e-42  0.0019394066  04_Early.Baso FOXA1
#> 5  6.144314e-242 2.457726e-241  0.0021416815  05_CMP.LMPP FOXA1
#> 6  9.919074e-102 3.967629e-101  0.0022356699  06_CLP.1 FOXA1
```

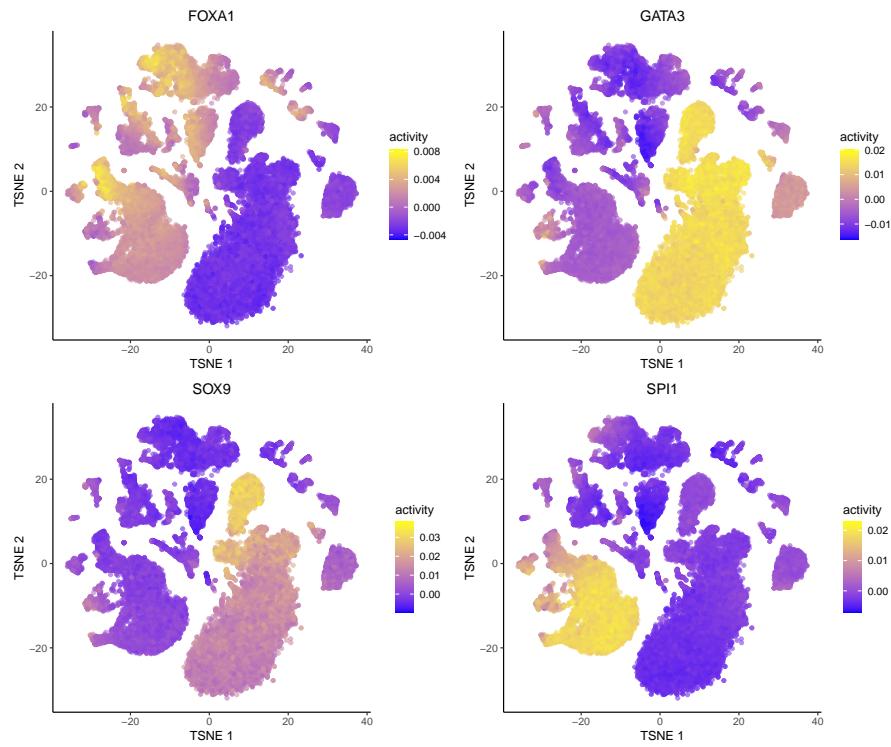
## 4.7 Visualizing TF activities

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 = T)
```

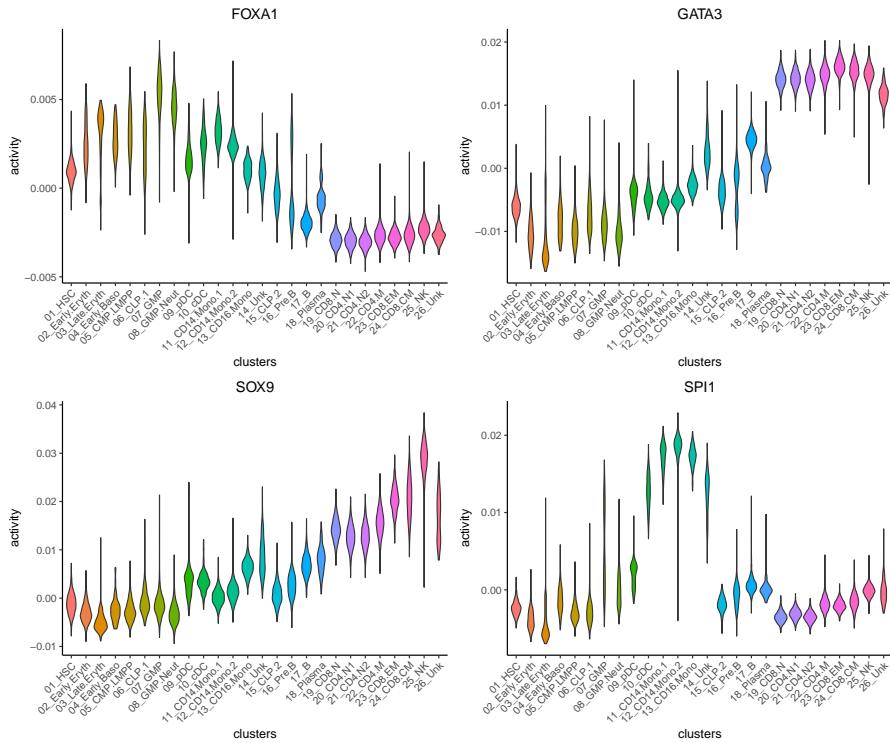
## Hematopoiesis tutorial - ArchR project



Violin plots:

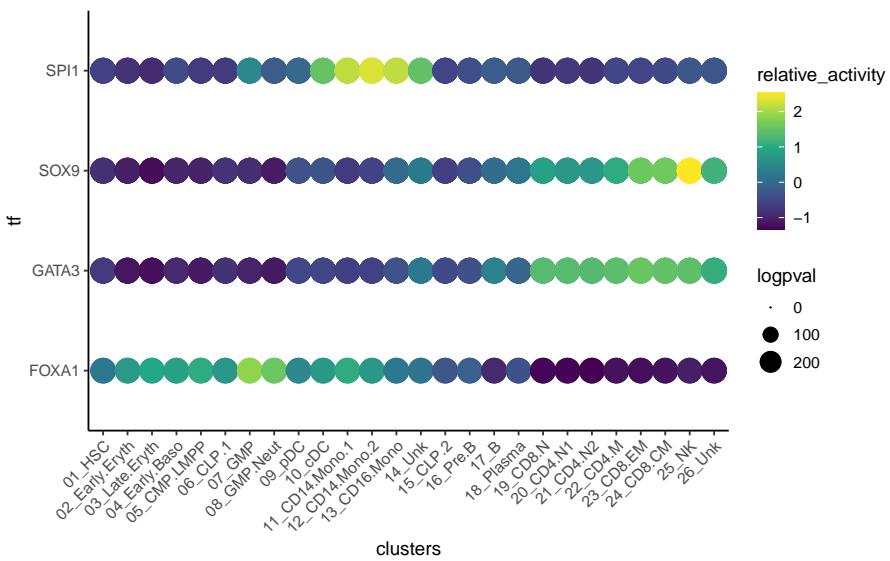
```
plotActivityViolin(activity_matrix = score.combine,
                    tf = c("FOXA1", "GATA3", "SOX9", "SPI1"),
                    clusters = sce$BioClassification)
```

## Hematopoiesis tutorial - ArchR project



Bubble plot:

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



## 5 Session Info

```

sessionInfo()
#> R version 4.2.0 (2022-04-22)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 18.04.6 LTS
#>
#> Matrix products: default
#> BLAS: /usr/local/lib/R/lib/libRblas.so
#> LAPACK: /usr/local/lib/R/lib/libRlapack.so
#>
#> Random number generation:
#> RNG: L'Ecuyer-CMRG
#> Normal: Inversion
#> Sample: Rejection
#>
#> locale:
#> [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
#> [3] LC_TIME=en_US.UTF-8       LC_COLLATE=en_US.UTF-8
#> [5] LC_MONETARY=en_US.UTF-8   LC_MESSAGES=en_US.UTF-8
#> [7] LC_PAPER=en_US.UTF-8     LC_NAME=C
#> [9] LC_ADDRESS=C             LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] parallel grid      stats4    stats      graphics grDevices utils
#> [8] datasets methods   base
#>
#> other attached packages:
#> [1] nabor_0.5.0            rhdf5_2.42.0
#> [3] Rcpp_1.0.9              Matrix_1.5-3
#> [5] data.table_1.14.6       stringr_1.4.0
#> [7] plyr_1.8.8              magrittr_2.0.3
#> [9] ggplot2_3.4.0           gtable_0.3.1
#> [11] gtools_3.9.3           gridExtra_2.3
#> [13] ArchR_1.0.2            epiregulon_1.0.18
#> [15] SingleCellExperiment_1.20.0 SummarizedExperiment_1.29.1
#> [17] Biobase_2.58.0          GenomicRanges_1.50.1
#> [19] GenomeInfoDb_1.34.3     IRanges_2.32.0
#> [21] S4Vectors_0.36.0         BiocGenerics_0.44.0
#> [23] MatrixGenerics_1.10.0   matrixStats_0.62.0
#> [25] BiocStyle_2.26.0        rmarkdown_2.18
#>
#> loaded via a namespace (and not attached):
#> [1] backports_1.4.1           igraph_1.3.5
#> [3] GSEABase_1.60.0           BiocParallel_1.32.1
#> [5] scater_1.26.1            digest_0.6.29
#> [7] htmltools_0.5.3           viridis_0.6.2
#> [9] fansi_1.0.3               checkmate_2.1.0
#> [11] memoise_2.0.1            base64url_1.4

```

## Hematopoiesis tutorial - ArchR project

```
#> [13] ScaledMatrix_1.6.0           cluster_2.1.3
#> [15] limma_3.54.0                Biostrings_2.66.0
#> [17] annotate_1.76.0              colorspace_2.0-3
#> [19] blob_1.2.3                  ggrepel_0.9.2
#> [21] xfun_0.31                   dplyr_1.0.10
#> [23] crayon_1.5.1                RCurl_1.98-1.9
#> [25] jsonlite_1.8.3              graph_1.76.0
#> [27] glue_1.6.2                  zlibbioc_1.44.0
#> [29] XVector_0.38.0              DelayedArray_0.24.0
#> [31] gp.cache_1.7.1              genomitory.schemas_0.99.0
#> [33] BiocSingular_1.14.0          Rhdf5lib_1.20.0
#> [35] HDF5Array_1.26.0             scales_1.2.1
#> [37] DBI_1.1.3                  edgeR_3.40.0
#> [39] viridisLite_0.4.1            xtable_1.8-4
#> [41] qrng_0.3.0                  bit_4.0.5
#> [43] rsvd_1.0.5                  GSVA_1.46.0
#> [45] metapod_1.6.0               httr_1.4.3
#> [47] artificer.ranges_1.3.4       pkgconfig_2.0.3
#> [49] XML_3.99-0.12                farver_2.1.1
#> [51] scuttle_1.8.0                locfit_1.5-9.6
#> [53] utf8_1.2.2                  tidyselect_1.2.0
#> [55] labeling_0.4.2              rlang_1.0.6
#> [57] reshape2_1.4.4              AnnotationDbi_1.60.0
#> [59] munsell_0.5.0                tools_4.2.0
#> [61] cachem_1.0.6                cli_3.4.1
#> [63] generics_0.1.3              RSQLite_2.2.18
#> [65] evaluate_0.18                fastmap_1.1.0
#> [67] yaml_2.3.5                  knitr_1.40
#> [69] bit64_4.0.5                artificer.schemas_0.99.2
#> [71] KEGGREST_1.38.0              sparseMatrixStats_1.10.0
#> [73] artificer.base_1.3.19        scran_1.26.0
#> [75] compiler_4.2.0                beeswarm_0.4.0
#> [77] filelock_1.0.2              png_0.1-7
#> [79] tibble_3.1.8                 statmod_1.4.37
#> [81] stringi_1.7.6                genomitory_2.1.6
#> [83] lattice_0.20-45             bluster_1.8.0
#> [85] vctrs_0.5.1                 pillar_1.8.1
#> [87] metacommons_1.9.0            lifecycle_1.0.3
#> [89] rhdf5filters_1.10.0          BiocManager_1.30.19
#> [91] BiocNeighbors_1.16.0          cowplot_1.1.1
#> [93] bitops_1.0-7                 irlba_2.3.5.1
#> [95] patchwork_1.1.2              R6_2.5.1
#> [97] bookdown_0.30                gp.version_1.5.0
#> [99] viper_0.4.5                 codetools_0.2-18
#> [101] assertthat_0.2.1            gp.auth_1.7.0
#> [103] withr_2.5.0                GenomeInfoDbData_1.2.9
#> [105] beachmat_2.14.0             DelayedMatrixStats_1.20.0
#> [107] ArtifactoryDB_1.9.5         Cairo_1.6-0
#> [109] getPass_0.2-2              ggbeeswarm_0.6.0
```