Hematopoeisis tutorial - ArchR project

Shang-Yang and Xiaosai Yao

23 May 2023

Package

epiregulon 1.0.23

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

```
library(epiregulon)
#> Warning: multiple methods tables found for 'acbind'
#> Warning: multiple methods tables found for 'arbind'
```

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 : Version 1.0.3
#> For more information see our website : www.ArchRProject.com
#> If you encounter a bug please report : https://github.com/GreenleafLab/ArchR/issues
#> Loading Required Packages...
#>  Loading Package : devtools v2.4.5
#>  Loading Package : grid v4.3.0
#>  Loading Package : gridExtra v2.3
#>  Loading Package : gtools v3.9.4
#>  Loading Package : gtable v0.3.3
#>  Loading Package : ggplot2 v3.4.2
#>  Loading Package : magrittr v2.0.3
#>  Loading Package : plyr v1.8.8
#>  Loading Package : stringr v1.5.0
#>  Loading Package : data.table v1.14.8
#>  Loading Package : matrixStats v0.63.0
#>  Loading Package : sparseMatrixStats v1.11.1
#>  Loading Package : S4Vectors v0.38.0
#>  Loading Package : GenomicRanges v1.52.0
#>  Loading Package : BiocGenerics v0.46.0
#>  Loading Package : Matrix v1.5.3
#>  Loading Package : Rcpp v1.0.10
#>  Loading Package : RcppArmadillo v0.12.2.0.0
```

```
#>  Loading Package : SummarizedExperiment v1.29.1
#>  Loading Package : rhdf5 v2.44.0
archR_project_path <- "/gstore/project/lineage/sam/heme_GRN/OUTPUT"
proj <- loadArchRProject(path = archR_project_path, showLogo = FALSE)
#> Successfully loaded ArchRProject!
getAvailableMatrices(proj)
#> [1] "GeneIntegrationMatrix" "GeneScoreMatrix"       "PeakMatrix"
#> [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")
#> see ?scMultiome and browseVignettes('scMultiome') for documentation
#> loading from cache
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...
#> 2023-05-23 12:14:02.861078 : Getting Available Matrices, 0 mins elapsed.
#> 2023-05-23 12:14:03.161726 : Filtered Low Prediction Score Cells (684 of 10250, 0.067), 0.001 mins elapsed
#> 2023-05-23 12:14:04.42426 : Computing KNN, 0.022 mins elapsed.
#> 2023-05-23 12:14:05.020204 : Identifying Non-Overlapping KNN pairs, 0.032 mins elapsed.
#> 2023-05-23 12:14:07.246677 : Identified 494 Groupings!, 0.069 mins elapsed.
#> 2023-05-23 12:14:07.294926 : Getting Group RNA Matrix, 0.07 mins elapsed.
#> 2023-05-23 12:15:33.496547 : Getting Group ATAC Matrix, 1.507 mins elapsed.
#> 2023-05-23 12:16:50.725463 : Normalizing Group Matrices, 2.794 mins elapsed.
#> 2023-05-23 12:16:56.37164 : Finding Peak Gene Pairings, 2.888 mins elapsed.
#> 2023-05-23 12:16:56.842923 : Computing Correlations, 2.896 mins elapsed.
#> 2023-05-23 12:17:03.376587 : Completed Peak2Gene Correlations!, 3.005 mins elapsed.
head(p2g)
#> DataFrame with 6 rows and 8 columns
#>     idxATAC      chr     start       end    idxRNA      target Correlation
#>   <integer> <factor> <integer> <integer> <integer> <character>   <numeric>
#> 1         6     chr1    779906    780406         9       ISG15    0.530835
#> 2        24     chr1    894453    894953         6      KLHL17    0.569856
#> 3        24     chr1    894453    894953        14    TNFRSF18    0.565133
#> 4        25     chr1    894960    895460         9       ISG15    0.589576
#> 5        25     chr1    894960    895460        14    TNFRSF18    0.568077
#> 6        37     chr1    935289    935789         8        HES4    0.667962
#>    distance
#>   <numeric>
#> 1    168691
#> 2      1264
#> 3    247386
#> 4     53637
#> 5    246879
#> 6        13
```

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
#> 1005       6    94 BRD4
#> 1006       6   137 CDK9
#> 1007       6   267  ERG
#> 1008       6   272 ETS1
#> 1009       6   291 FLI1
#> 1010       6   395 HIC1
```

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)
#> DataFrame with 6 rows and 10 columns
#>     idxATAC      chr     start       end    idxRNA      target     corr
#>   <integer> <factor> <integer> <integer> <integer> <character> <matrix>
#> 1         6     chr1    779906    780406         9       ISG15 0.530835
#> 2         6     chr1    779906    780406         9       ISG15 0.530835
#> 3         6     chr1    779906    780406         9       ISG15 0.530835
#> 4         6     chr1    779906    780406         9       ISG15 0.530835
#> 5         6     chr1    779906    780406         9       ISG15 0.530835
#> 6         6     chr1    779906    780406         9       ISG15 0.530835
#>    distance     idxTF          tf
#>   <numeric> <integer> <character>
#> 1    168691        94        BRD4
#> 2    168691       137        CDK9
#> 3    168691       267         ERG
#> 4    168691       272        ETS1
#> 5    168691       291        FLI1
#> 6    168691       395        HIC1
```

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] 47805
```

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"
)
#> 2023-05-23 12:18:55.047105 : Organizing colData, 0.885 mins elapsed.
#> 2023-05-23 12:18:55.138353 : Organizing rowData, 0.886 mins elapsed.
#> 2023-05-23 12:18:55.14009 : Organizing rowRanges, 0.886 mins elapsed.
#> 2023-05-23 12:18:55.144174 : Organizing Assays (1 of 1), 0.886 mins elapsed.
#> 2023-05-23 12:18:59.256786 : Constructing SummarizedExperiment, 0.955 mins elapsed.
#> 2023-05-23 12:19:00.496464 : Finished Matrix Creation, 0.976 mins elapsed.

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

PeakMatrix <- getMatrixFromProject(
    ArchRProj = proj,
    useMatrix = "PeakMatrix",
    useSeqnames = NULL,
    verbose = TRUE,
    binarize = FALSE,
    threads = 1,
    logFile = "x"
)
#> 2023-05-23 12:19:15.053312 : Organizing colData, 0.243 mins elapsed.
#> 2023-05-23 12:19:15.143316 : Organizing rowData, 0.244 mins elapsed.
#> 2023-05-23 12:19:15.148525 : Organizing rowRanges, 0.244 mins elapsed.
#> 2023-05-23 12:19:15.1566 : Organizing Assays (1 of 1), 0.244 mins elapsed.
#> 2023-05-23 12:19:15.772872 : Constructing SummarizedExperiment, 0.254 mins elapsed.
#> 2023-05-23 12:19:20.393713 : Finished Matrix Creation, 0.332 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 chi.sq tests using a regulon cutoff of pval<0.05
#> pruning regulons
```

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)
#> DataFrame with 6 rows and 11 columns
#>     idxATAC      chr     start       end    idxRNA      target     corr
#>   <integer> <factor> <integer> <integer> <integer> <character> <matrix>
#> 1        24     chr1    894453    894953         6      KLHL17 0.569856
#> 2        24     chr1    894453    894953        14    TNFRSF18 0.565133
#> 3        37     chr1    935289    935789         8        HES4 0.667962
#> 4        42     chr1    948574    949074        15     TNFRSF4 0.540911
#> 5        42     chr1    948574    949074         9       ISG15 0.506503
#> 6        42     chr1    948574    949074        14    TNFRSF18 0.530413
#>    distance     idxTF          tf     weight
#>   <numeric> <integer> <character>  <numeric>
#> 1      1264       296       FOXA1 -0.1063868
#> 2    247386       296       FOXA1 -0.0752085
#> 3        13       296       FOXA1 -0.0462854
#> 4    200724       296       FOXA1 -0.0780133
#> 5        23       296       FOXA1 -0.1515103
#> 6    193265       296       FOXA1 -0.0752085
```

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")
#> calculating TF activity from regulon using weightedmean
#> aggregating regulons...
#> creating weight matrix...
#> calculating activity scores...
#> normalize by the number of targets...
head(score.combine[,1:10])
#> 4 x 10 sparse Matrix of class "dgCMatrix"
#>   [[ suppressing 10 column names 'CD34_32_R5:AAACCTGAGTATCGAA-1', 'CD34_32_R5:AAACCTGAGTCGTTTG-1', 'CD34_3
#>
#> FOXA1  0.006637545  0.014247842  0.0018420894  0.004645328  0.0024000851
#> GATA3 -0.020414877 -0.027550050 -0.0148136253 -0.017496930 -0.0123770694
#> SOX9  -0.001866304 -0.006585295 -0.0009248676 -0.002364701 -0.0003773008
#> SPI1  -0.006862495  0.005455083 -0.0065180209 -0.008583923 -0.0045408081
#>
#> FOXA1  0.001393019 -0.001681288  0.001476732  0.003774192  0.003676724
#> GATA3 -0.011442372 -0.005501289 -0.012252541 -0.022089690 -0.017693718
#> SOX9   0.001152362  0.002891448  0.000686534 -0.002155383 -0.003717816
#> SPI1  -0.008117866 -0.005715629 -0.005647495 -0.011645509 -0.005537353
```

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.

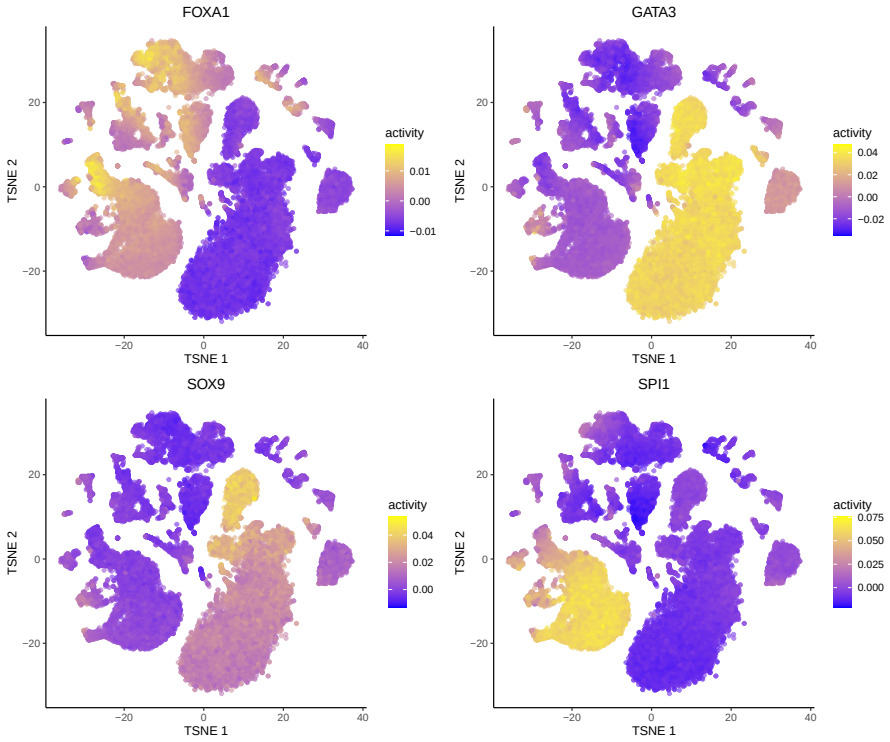
```
markers <- getSigGenes(da_list, fdr_cutoff = 0.05)
#> Using a logFC cutoff of 0 for class 01_HSC
#> Using a logFC cutoff of 0 for class 02_Early.Eryth
#> Using a logFC cutoff of 0 for class 03_Late.Eryth
#> Using a logFC cutoff of 0 for class 04_Early.Baso
#> Using a logFC cutoff of 0 for class 05_CMP.LMPP
#> 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 0.1 for class 11_CD14.Mono.1
#> Using a logFC cutoff of 0.1 for class 12_CD14.Mono.2
#> Using a logFC cutoff of 0.1 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  1.283895e-133 5.135582e-133   0.003069588 02_Early.Eryth FOXA1
#> 2  1.105248e-105 4.420990e-105   0.004793624  03_Late.Eryth FOXA1
#> 3   3.577504e-43  1.431002e-42   0.004411075  04_Early.Baso FOXA1
#> 4  4.049189e-226 1.619676e-225   0.004706581    05_CMP.LMPP FOXA1
#> 5  3.534306e-105 1.413722e-104   0.005113709       06_CLP.1 FOXA1
#> 21  0.000000e+00  0.000000e+00   0.020074219         07_GMP  SPI1
```

4.7 Visualizing TF activities

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

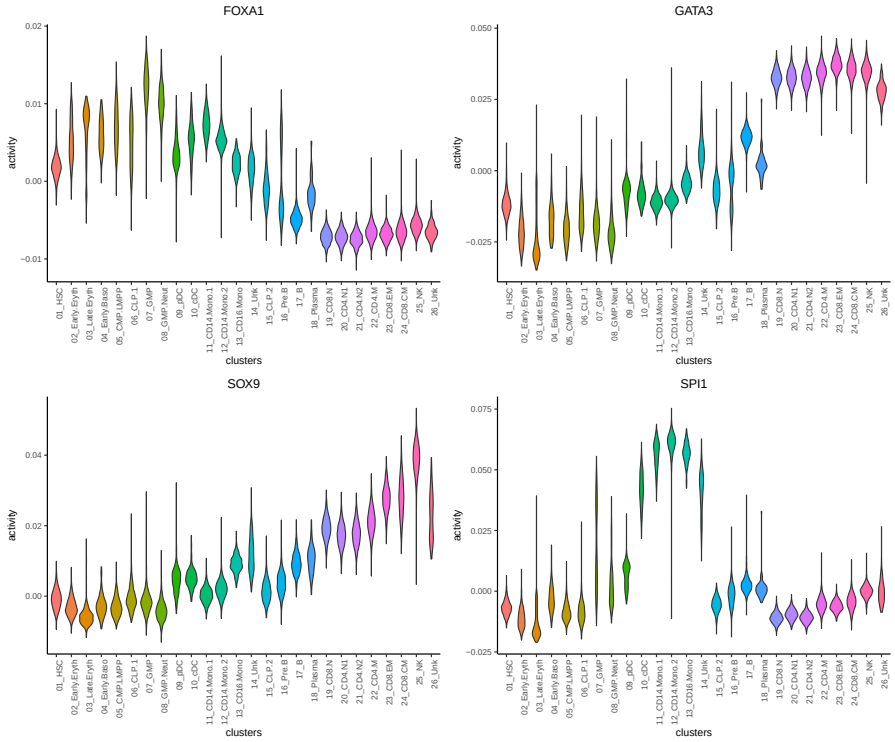
tSNE or UMAP plots:

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



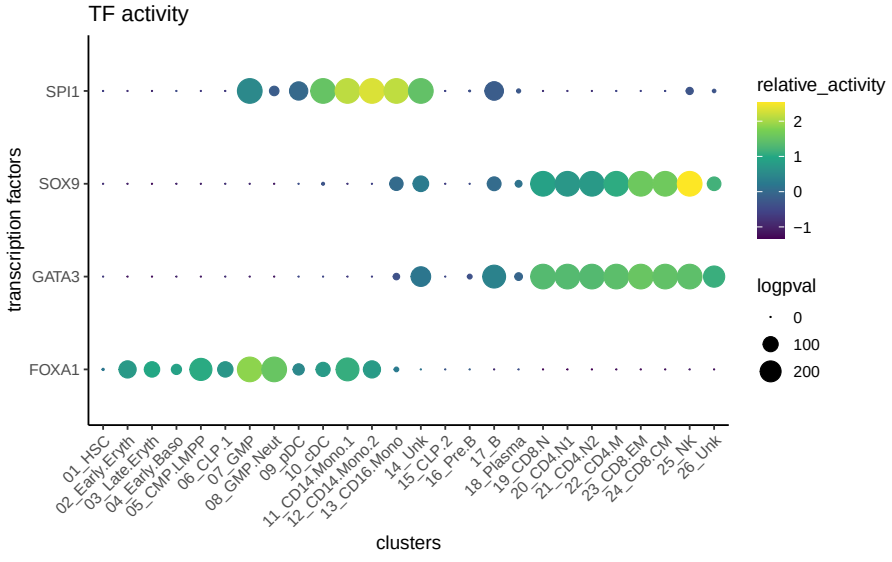
Violin plots:

```
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()
#> R Under development (unstable) (2022-11-21 r83371)
#> 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
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C         LC_TIME=C
#>  [4] LC_COLLATE=C         LC_MONETARY=C        LC_MESSAGES=C
#>  [7] LC_PAPER=C           LC_NAME=C            LC_ADDRESS=C
#> [10] LC_TELEPHONE=C       LC_MEASUREMENT=C     LC_IDENTIFICATION=C
#>
#> time zone: Etc/UTC
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] grid      stats4    stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] nabor_0.5.0                  scMultiome_1.1.0
#>  [3] MultiAssayExperiment_1.25.11 ExperimentHub_2.7.1
#>  [5] AnnotationHub_3.7.4          BiocFileCache_2.8.0
#>  [7] dbplyr_2.3.2                 rhdf5_2.44.0
#>  [9] RcppArmadillo_0.12.2.0.0     Rcpp_1.0.10
#> [11] Matrix_1.5-3                 sparseMatrixStats_1.11.1
#> [13] data.table_1.14.8            stringr_1.5.0
#> [15] plyr_1.8.8                   magrittr_2.0.3
#> [17] ggplot2_3.4.2                gtable_0.3.3
#> [19] gtools_3.9.4                 gridExtra_2.3
#> [21] devtools_2.4.5               usethis_2.1.6
#> [23] ArchR_1.0.3                  epiregulon_1.0.23
#> [25] SingleCellExperiment_1.21.1  SummarizedExperiment_1.29.1
#> [27] Biobase_2.60.0               GenomicRanges_1.52.0
#> [29] GenomeInfoDb_1.36.0          IRanges_2.34.0
#> [31] S4Vectors_0.38.0             BiocGenerics_0.46.0
#> [33] MatrixGenerics_1.12.0        matrixStats_0.63.0
#> [35] BiocStyle_2.27.2
#>
#> loaded via a namespace (and not attached):
#>   [1] later_1.3.0                   bitops_1.0-7
#>   [3] filelock_1.0.2                tibble_3.2.1
#>   [5] graph_1.77.3                  XML_3.99-0.14
#>   [7] lifecycle_1.0.3               edgeR_3.41.9
#>   [9] processx_3.8.0                lattice_0.20-45
```

```
#>  [11] backports_1.4.1               limma_3.55.10
#>  [13] rmarkdown_2.21                yaml_2.3.7
#>  [15] remotes_2.4.2                 metapod_1.7.0
#>  [17] httpuv_1.6.9                  sessioninfo_1.2.2
#>  [19] pkgbuild_1.4.0                cowplot_1.1.1
#>  [21] DBI_1.1.3                     pkgload_1.3.2
#>  [23] zlibbioc_1.46.0               purrr_1.0.1
#>  [25] RCurl_1.98-1.12               rappdirs_0.3.3
#>  [27] GenomeInfoDbData_1.2.10       ggrepel_0.9.3
#>  [29] irlba_2.3.5.1                 GSVA_1.47.3
#>  [31] annotate_1.77.0               dqrng_0.3.0
#>  [33] DelayedMatrixStats_1.21.0     codetools_0.2-18
#>  [35] DelayedArray_0.26.2           scuttle_1.9.4
#>  [37] tidyselect_1.2.0              farver_2.1.1
#>  [39] viridis_0.6.3                 ScaledMatrix_1.7.1
#>  [41] BiocNeighbors_1.17.1          ellipsis_0.3.2
#>  [43] scater_1.27.9                 tools_4.3.0
#>  [45] glue_1.6.2                    xfun_0.39
#>  [47] dplyr_1.1.2                   HDF5Array_1.28.1
#>  [49] withr_2.5.0                   BiocManager_1.30.20
#>  [51] fastmap_1.1.1                 rhdf5filters_1.11.2
#>  [53] bluster_1.9.1                 fansi_1.0.4
#>  [55] callr_3.7.3                   digest_0.6.31
#>  [57] rsvd_1.0.5                    R6_2.5.1
#>  [59] mime_0.12                     colorspace_2.1-0
#>  [61] RSQLite_2.3.1                 utf8_1.2.3
#>  [63] generics_0.1.3                prettyunits_1.1.1
#>  [65] httr_1.4.5                    htmlwidgets_1.6.2
#>  [67] S4Arrays_1.0.1                pkgconfig_2.0.3
#>  [69] blob_1.2.4                    XVector_0.40.0
#>  [71] htmltools_0.5.5               profvis_0.3.7
#>  [73] bookdown_0.33                 GSEABase_1.61.2
#>  [75] scales_1.2.1                  png_0.1-8
#>  [77] scran_1.27.3                  knitr_1.42
#>  [79] rstudioapi_0.14               reshape2_1.4.4
#>  [81] checkmate_2.2.0               curl_5.0.0
#>  [83] cachem_1.0.8                  BiocVersion_3.17.1
#>  [85] parallel_4.3.0                miniUI_0.1.1.1
#>  [87] vipor_0.4.5                   AnnotationDbi_1.62.0
#>  [89] pillar_1.9.0                  vctrs_0.6.2
#>  [91] urlchecker_1.0.1              promises_1.2.0.1
#>  [93] BiocSingular_1.15.0           beachmat_2.15.2
#>  [95] xtable_1.8-4                  cluster_2.1.4
#>  [97] beeswarm_0.4.0                evaluate_0.20
#>  [99] cli_3.6.1                     locfit_1.5-9.7
#> [101] compiler_4.3.0                rlang_1.1.1
#> [103] crayon_1.5.2                  labeling_0.4.2
#> [105] ps_1.7.2                      fs_1.6.2
#> [107] ggbeeswarm_0.7.1              stringi_1.7.12
#> [109] viridisLite_0.4.2             BiocParallel_1.34.0
#> [111] munsell_0.5.0                 Biostrings_2.68.0
```

```
#> [113] patchwork_1.1.2              bit64_4.0.5
#> [115] Rhdf5lib_1.21.1              KEGGREST_1.40.0
#> [117] statmod_1.5.0                shiny_1.7.4
#> [119] interactiveDisplayBase_1.37.0 igraph_1.4.2
#> [121] memoise_2.0.1                bit_4.0.5
```