Epiregulon

true true

February 15th, 2022

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.

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.

Installation

```
#devtools::install_github("OncoBioinfo-MO/Epiregulon", host = "https://github.roche.com/api/v3")
library(epiregulon)
```

Data preparation

Please refer to the full ArchR [manual] (https://www.archrproject.com/bookdown/index.html) for instructions

Before running Epiregulon, the following analyses need to be completed: 1. Obtain a peak matrix on scATACseq by using addGroupCoverages > addReproduciblePeakSet > addPeakMatrix. See chapter 10 from ArchR manual 2. RNAseq integration. a. For unpaired scATACseq, use addGeneIntegrationMatrix. See chapter [8] (https://www.archrproject.com/bookdown/defining-cluster-identity-with-scrna-seq.html) from ArchR manual b. For multiome data, use addGeneExpressionMatrix. See multiome tutorial 3. Perform dimensionality reduction from with either single modalities or joint scRNAseq and scATACseq using addCombinedDims

To verify that all the necessary matrices are present,

Quick start

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 are available. We plan to add options for murine genomes and perhaps other model organisms in the near future.

```
grl <- getTFMotifInfo(genome="hg19")
head(grl)
#> GRangesList object of length 6:
#> $ADNP
#> GRanges object with 20474 ranges and 0 metadata columns:
#>           seqnames              ranges strand
#>              <Rle>           <IRanges>  <Rle>
#>       [1]     chr1       565199-565456      *
#>       [2]     chr1       569272-569544      *
#>       [3]     chr1       895823-896145      *
#>       [4]     chr1       946692-947524      *
#>       [5]     chr1       993405-993727      *
#>       ...      ...                 ...    ...
#>   [20470]     chrX 153402026-153403267      *
#>   [20471]     chrX 153729859-153730943      *
#>   [20472]     chrX 153979887-153980719      *
#>   [20473]     chrX 154297459-154298291      *
#>   [20474]     chrX 154446717-154447848      *
#>   -------
#>   seqinfo: 52 sequences from an unspecified genome; no seqlengths
#>
#> ...
#> <5 more elements>
```

Link ATACseq peaks to target genes

Next, we compute peak to gene correlations using the addPeak2GeneLinks function from ArchR package. The user would need to supply a path to an ArchR directory with a project contains peak and gene matrices, as well as already performed Latent semantic indexing (LSI) dimensionality reduction and integration with a scRNA-seq dataset. The example project shown here utilizes the tutorial datasets provided by the ArchR development team.

```
# path to ArchR project
p2g <- getP2Glinks(archR_project_path)
#> Setting ArchRLogging = FALSE
#> 2022-02-17 15:16:36 : Getting Available Matrices, 0 mins elapsed.
#> 2022-02-17 15:16:36 : Filtered Low Prediction Score Cells (684 of 10250, 0.067), 0.001 mins elapsed.
#> 2022-02-17 15:16:36 : Computing KNN, 0.004 mins elapsed.
#> 2022-02-17 15:16:56 : Identifying Non-Overlapping KNN pairs, 0.334 mins elapsed.
#> 2022-02-17 15:16:59 : Identified 494 Groupings!, 0.377 mins elapsed.
#> 2022-02-17 15:16:59 : Getting Group RNA Matrix, 0.382 mins elapsed.
#> 2022-02-17 15:18:14 : Getting Group ATAC Matrix, 1.639 mins elapsed.
#> 2022-02-17 15:19:15 : Normalizing Group Matrices, 2.641 mins elapsed.
#> 2022-02-17 15:19:19 : Finding Peak Gene Pairings, 2.712 mins elapsed.
#> 2022-02-17 15:19:19 : Computing Correlations, 2.719 mins elapsed.
```

```
#> 2022-02-17 15:19:26 : Completed Peak2Gene Correlations!, 2.834 mins elapsed.
head(p2g)
#>   idxATAC Chrom idxRNA     Gene Correlation
#> 1       6  chr1      9    ISG15   0.5308352
#> 3      24  chr1      6   KLHL17   0.5698560
#> 2      24  chr1     14 TNFRSF18   0.5651328
#> 5      25  chr1      9    ISG15   0.5895757
#> 4      25  chr1     14 TNFRSF18   0.5680771
#> 6      37  chr1      8     HES4   0.6679620
```

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...
#> Sucess!
head(overlap)
#>      idxATAC idxTF     tf
#> 522        6   427    MAX
#> 523        6   601 POLR2G
#> 524        6   787   TAL1
#> 525        6   798  TCF12
#> 526        6   811   TERC
#> 2200      24     3   AFF1
```

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=TRUE)
head(regulon)
#>       tf  target      corr
#> 1   AFF1 A2M-AS1 0.6021363
#> 2 ARID4A A2M-AS1 0.6021363
#> 3 ARID4B A2M-AS1 0.6021363
#> 4  ASH2L A2M-AS1 0.6021363
#> 5   ATF1 A2M-AS1 0.6021363
#> 6   ATF2 A2M-AS1 0.6021363
```

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 %>% dplyr::filter(tf %in% TFs)
nrow(regulon)
#> [1] 15336

regulon.w=addWeights(regulon=regulon,
                     sce=sce,
                     cluster_factor="BioClassification",
                     block_factor=NULL,
                     corr=TRUE,
                     MI=FALSE,
                     multicore=TRUE)
#> calculating average expression across clusters...
#> computing correlation of the regulon...
#>   |                                                                              |
head(regulon.w)
#>       tf  target      corr     weight
#> 1  GATA3 A2M-AS1 0.5683129  0.6487928
#> 4  GATA3  A4GALT 0.7320947 -0.1683055
#> 9  GATA3    AAK1 0.6251292  0.8074204
#> 12 GATA3   AAMDC 0.5667211 -0.4027895
#> 14 GATA3    AAMP 0.5984585 -0.3128506
#> 17 GATA3    AAR2 0.6434904 -0.2774245
```

Calculate TF activity

Finally, the activities for a specific TF in each cell are computed by averaging expressions of target genes linked to the TF weighted by the correlation variable of user’s choice.

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

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} $corr_i$ is the weight of TF and target i

```
score.combine <- calculateActivity(sce, regulon.w, "weight", method="weightedMean", assay = "logcounts")
#> weightedmean
#> calculating TF activity from regulon using weightedmean
#>   |                                                                              |
head(score.combine[,1:10])
#>       CD34_32_R5:AAACCTGAGTATCGAA-1 CD34_32_R5:AAACCTGAGTCGTTTG-1
#> GATA3                   -0.021012397                  -0.027471145
#> SPI1                    -0.003549379                   0.010048047
#> FOXA1                    0.006467344                   0.014336410
#> SOX9                     0.002726760                  -0.001256817
#>       CD34_32_R5:AAACCTGGTTCCACAA-1 CD34_32_R5:AAACGGGAGCTTCGCG-1
```

```
#> GATA3                   -0.016458312                   -0.0165527279
#> SPI1                    -0.004201749                   -0.0051176003
#> FOXA1                    0.002326760                    0.0040956059
#> SOX9                     0.003799146                   -0.0007582545
#>       CD34_32_R5:AAACGGGAGGGAGTAA-1 CD34_32_R5:AAACGGGAGTTACGGG-1
#> GATA3                   -0.013892506                   -0.011573274
#> SPI1                    -0.001195408                   -0.005240490
#> FOXA1                    0.002576713                    0.001192130
#> SOX9                     0.007332733                    0.005315836
#>       CD34_32_R5:AAACGGGCAAGTAATG-1 CD34_32_R5:AAACGGGCAAGTTCTG-1
#> GATA3                   -0.005744173                   -0.012281146
#> SPI1                    -0.002644292                   -0.003329163
#> FOXA1                   -0.001796598                    0.002849936
#> SOX9                     0.006684796                    0.003803802
#>       CD34_32_R5:AAACGGGCACAGACAG-1 CD34_32_R5:AAACGGGGTAACGTTC-1
#> GATA3                   -0.024982442                   -0.0191374106
#> SPI1                    -0.008007367                   -0.0019797585
#> FOXA1                    0.003780386                    0.0032500098
#> SOX9                     0.001013378                    0.0004819318
```

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(score.combine, sce$BioClassification, pval.type="some", direction="
```

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, order = "summary.logFC", decreasing = T)
#> Using a logFC cutoff of 0 for class 1
#> Using a logFC cutoff of 0 for class 2
#> Using a logFC cutoff of 0 for class 3
#> Using a logFC cutoff of 0 for class 4
#> Using a logFC cutoff of 0 for class 5
#> Using a logFC cutoff of 0 for class 6
#> Using a logFC cutoff of 0 for class 7
#> Using a logFC cutoff of 0 for class 8
#> Using a logFC cutoff of 0 for class 9
#> Using a logFC cutoff of 0 for class 10
#> Using a logFC cutoff of 0.1 for class 11
#> Using a logFC cutoff of 0.1 for class 12
#> Using a logFC cutoff of 0.1 for class 13
#> Using a logFC cutoff of 0 for class 14
#> Using a logFC cutoff of 0 for class 15
#> Using a logFC cutoff of 0 for class 16
#> Using a logFC cutoff of 0 for class 17
#> Using a logFC cutoff of 0 for class 18
#> Using a logFC cutoff of 0 for class 19
#> Using a logFC cutoff of 0 for class 20
#> Using a logFC cutoff of 0 for class 21
#> Using a logFC cutoff of 0 for class 22
```

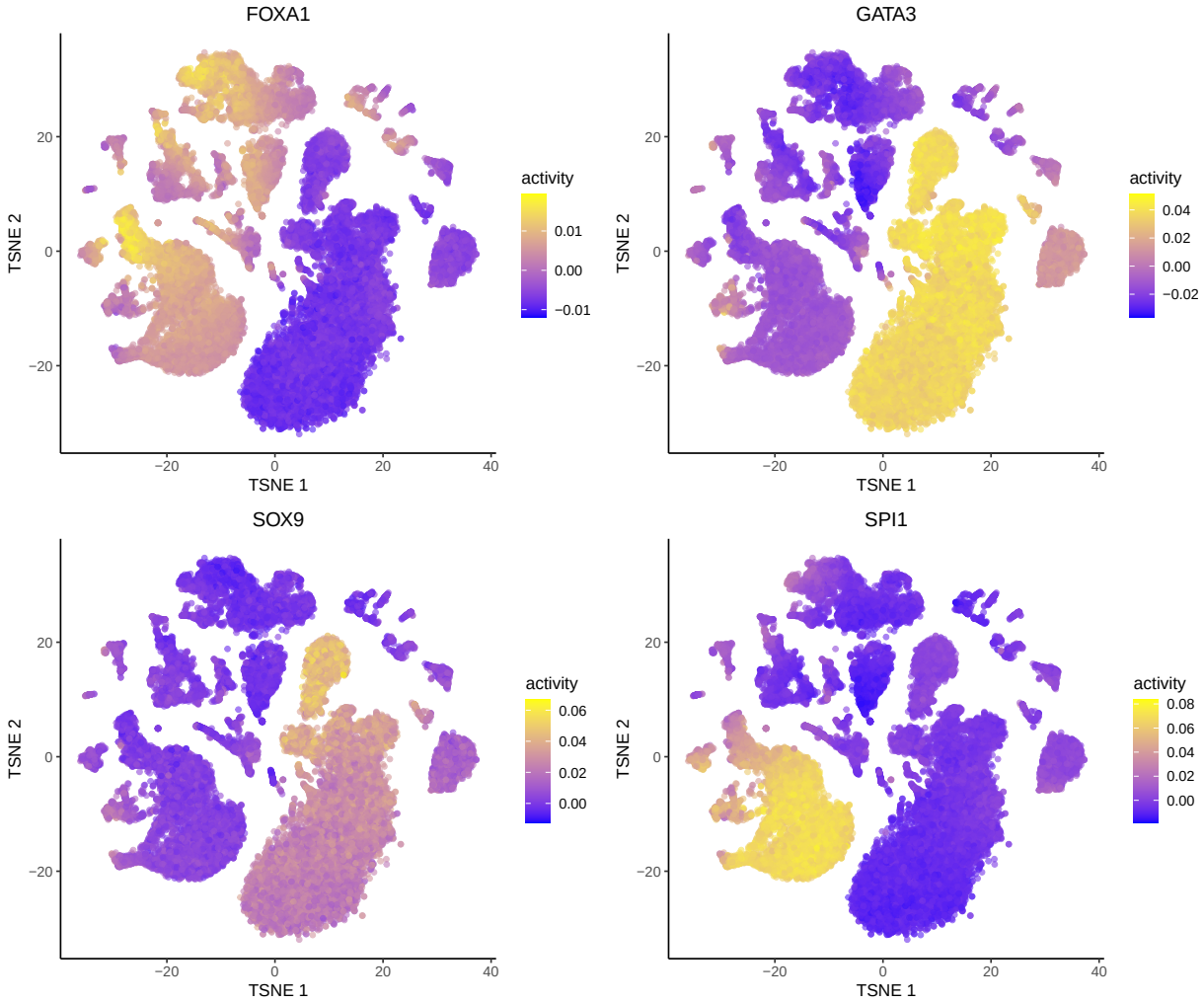
```
#> Using a logFC cutoff of 0 for class 23
#> Using a logFC cutoff of 0 for class 24
#> Using a logFC cutoff of 0 for class 25
#> Using a logFC cutoff of 0 for class 26
head(markers)
#>          p.value           FDR summary.logFC          class    tf
#> 1  2.196874e-123 8.787498e-123   0.002982791 02_Early.Eryth FOXA1
#> 2   1.089715e-84  4.358859e-84   0.004170094  03_Late.Eryth FOXA1
#> 3   5.227473e-37  2.090989e-36   0.004031504  04_Early.Baso FOXA1
#> 4  6.075948e-213 2.430379e-212   0.007563786    05_CMP.LMPP FOXA1
#> 5   6.748446e-62  2.699379e-61   0.002395315       06_CLP.1 FOXA1
#> 21  0.000000e+00  0.000000e+00   0.022055115         07_GMP  SPI1
```

Visualizing TF activities

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

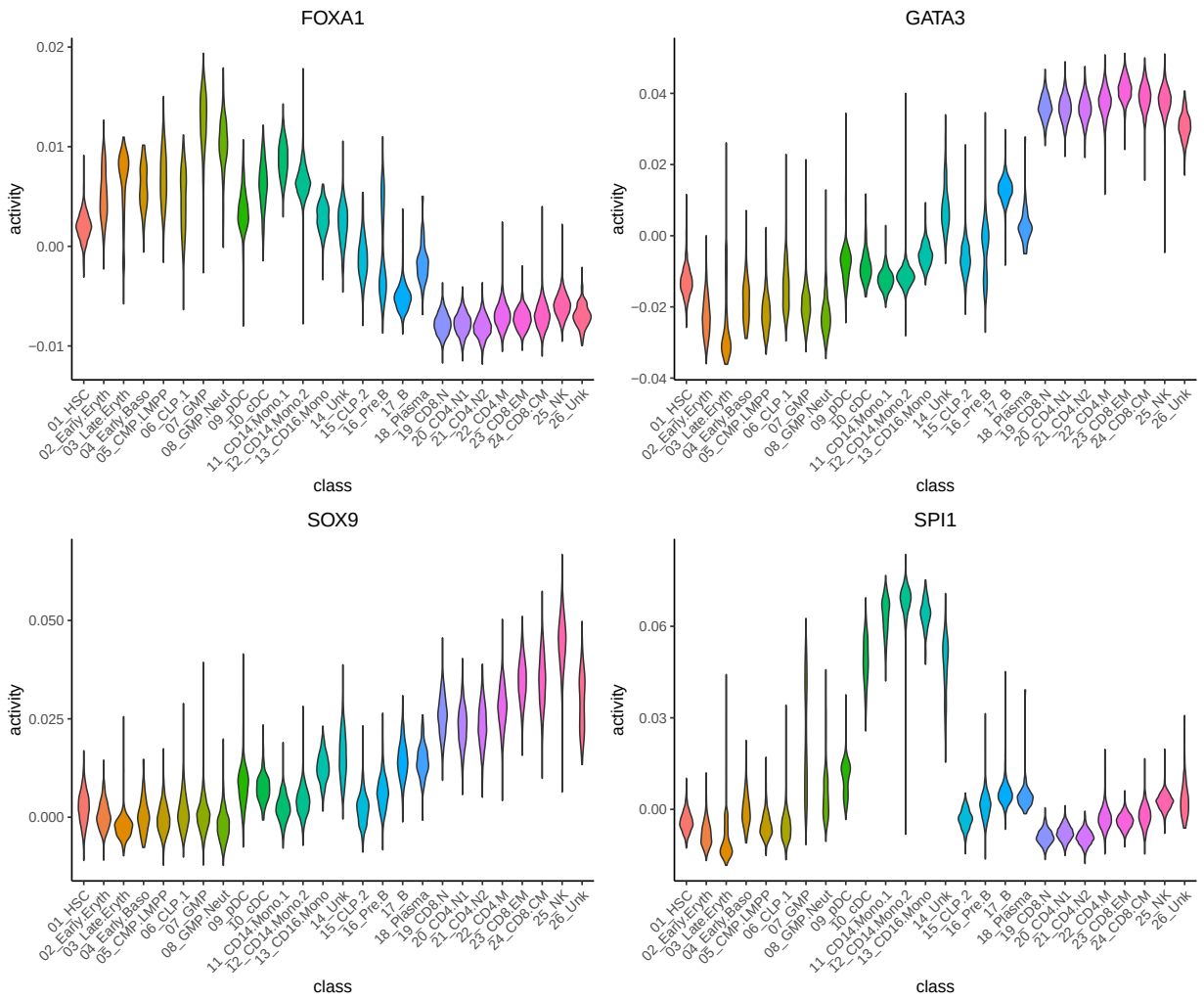
tSNE or UMAP plots:

```
plotActivityDim(sce, score.combine, c("FOXA1","GATA3","SOX9", "SPI1"), "TSNE", combine = T)
```



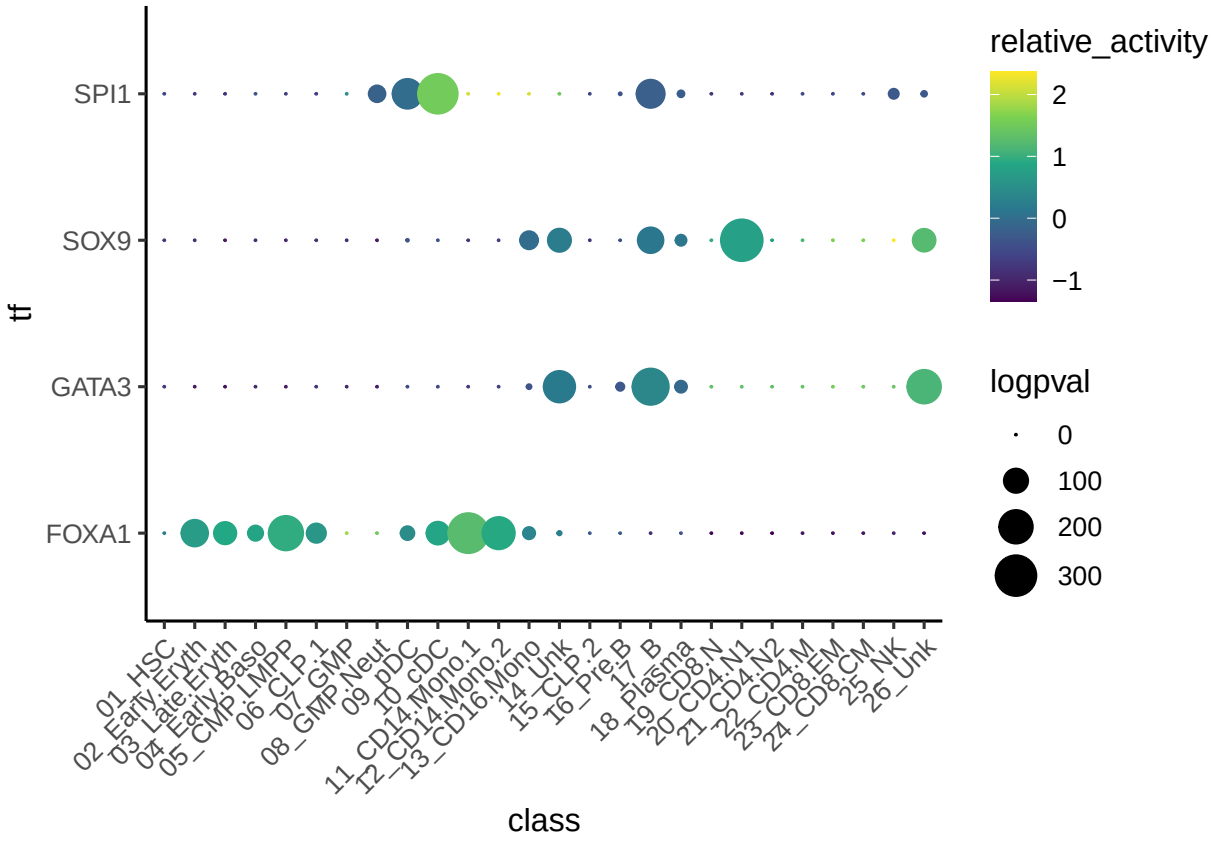
Violin plots:

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



Bubble plot:

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



Session Info

```
sessionInfo()
#> R version 4.1.0 (2021-05-18)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 18.04.5 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
#>  [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  stats4    stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
```

```
#> other attached packages:
#>  [1] nabor_0.5.0                 epiregulon_0.0.0.9000
#>  [3] ArchR_1.0.2                 magrittr_2.0.2
#>  [5] rhdf5_2.36.0                Matrix_1.3-4
#>  [7] data.table_1.14.0           SummarizedExperiment_1.22.0
#>  [9] Biobase_2.52.0              GenomicRanges_1.44.0
#> [11] GenomeInfoDb_1.28.2         IRanges_2.26.0
#> [13] S4Vectors_0.30.0            BiocGenerics_0.38.0
#> [15] MatrixGenerics_1.4.3        matrixStats_0.60.1
#> [17] ggplot2_3.3.5
#>
#> loaded via a namespace (and not attached):
#>   [1] ggbeeswarm_0.6.0            colorspace_2.0-2
#>   [3] ellipsis_0.3.2              scuttle_1.2.1
#>   [5] bluster_1.2.1               XVector_0.32.0
#>   [7] BiocNeighbors_1.10.0        farver_2.1.0
#>   [9] bit64_4.0.5                 fansi_1.0.2
#>  [11] codetools_0.2-18            sparseMatrixStats_1.4.2
#>  [13] cachem_1.0.6                knitr_1.33
#>  [15] scater_1.20.1               jsonlite_1.7.3
#>  [17] cluster_2.1.2               compiler_4.1.0
#>  [19] httr_1.4.2                  dqrng_0.3.0
#>  [21] backports_1.2.1             assertthat_0.2.1
#>  [23] fastmap_1.1.0               limma_3.48.3
#>  [25] BiocSingular_1.8.1          htmltools_0.5.2
#>  [27] tools_4.1.0                 rsvd_1.0.5
#>  [29] igraph_1.2.6                gp.auth_1.2.1
#>  [31] gtable_0.3.0                glue_1.6.1
#>  [33] GenomeInfoDbData_1.2.6      gp.cache_1.2.7
#>  [35] dplyr_1.0.7                 Rcpp_1.0.7
#>  [37] vctrs_0.3.8                 rhdf5filters_1.4.0
#>  [39] DelayedMatrixStats_1.14.3   xfun_0.29
#>  [41] stringr_1.4.0               beachmat_2.8.1
#>  [43] lifecycle_1.0.0             irlba_2.3.3
#>  [45] gtools_3.9.2                statmod_1.4.36
#>  [47] edgeR_3.34.0                getPass_0.2-2
#>  [49] zlibbioc_1.38.0             scales_1.1.1
#>  [51] SingleCellExperiment_1.14.1 yaml_2.2.2
#>  [53] memoise_2.0.0               gridExtra_2.3
#>  [55] metacommons_1.4.0           stringi_1.7.6
#>  [57] RSQLite_2.2.8               highr_0.9
#>  [59] ScaledMatrix_1.0.0          scran_1.20.1
#>  [61] filelock_1.0.2              BiocParallel_1.26.2
#>  [63] rlang_0.4.11                pkgconfig_2.0.3
#>  [65] bitops_1.0-7                evaluate_0.14
#>  [67] lattice_0.20-44             purrr_0.3.4
#>  [69] Rhdf5lib_1.14.2             labeling_0.4.2
#>  [71] patchwork_1.1.1             cowplot_1.1.1
#>  [73] bit_4.0.4                   tidyselect_1.1.1
#>  [75] R6_2.5.1                    generics_0.1.2
#>  [77] base64url_1.4               metapod_1.0.0
#>  [79] DelayedArray_0.18.0         DBI_1.1.1
#>  [81] pillar_1.6.2                withr_2.4.3
```

```
#>  [83] RCurl_1.98-1.4             tibble_3.1.4
#>  [85] crayon_1.5.0               utf8_1.2.2
#>  [87] rmarkdown_2.10             viridis_0.6.2
#>  [89] locfit_1.5-9.4             grid_4.1.0
#>  [91] genomitory_1.2.9           blob_1.2.2
#>  [93] gp.version_1.2.1           digest_0.6.29
#>  [95] tidyr_1.1.3                munsell_0.5.0
#>  [97] beeswarm_0.4.0             viridisLite_0.4.0
#>  [99] ArtifactDB_1.4.12          vipor_0.4.5
```