Multiome tutorial

Shang-Yang Chen* Xiaosai Yao†

February 4th, 2022

Introduction

This tutorial walks through an example of TF activity inference in single cell multiome data. This is a dataset generated by infecting LNCaP cells with NKX2-1 and GATA6 to examine the effects of these TFs on AR activity.

Installation

Epiregulon is currently available on R/dev

```
library(epiregulon)
```

Alternatively, you could install from gitlab

```
devtools::install_gitlab(repo='scwg/gene-transcriptional-network/activity-inference/Epiregulon',
                         auth_token = "<gitlab token>",
                         host = "https://code.roche.com/" )

library(epiregulon)
```

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 scATACseq 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 scRNAseq and scATACseq using addCombinedDims

Copy this ArchR project into your own directory

```
archR_project_path="/gstore/project/ar_ligands/NE/reprogram_seq/multiome_arrayed/OUTPUT/doubletremoved/
tutorial <- loadArchRProject(path = archR_project_path, showLogo = F)

# save tutorial data into your new directory and load it
```

*chens179@gene.com
†yaox19@gene.com

```
myarchR_project_path = "/gstore/scratch/u/yaox19/multiome"
saveArchRProject(ArchRProj = tutorial, outputDirectory = myarchR_project_path, load = F )
proj <- loadArchRProject(path = myarchR_project_path, showLogo = F)

setwd(myarchR_project_path)
```

We verify that "GeneExpressionMatrix" and "PeakMatrix" are present for this tutorial.

```
getAvailableMatrices(proj)
#> [1] "GeneExpressionMatrix" "GeneScoreMatrix"      "NEPCMatrix"
#> [4] "PeakMatrix"           "TileMatrix"
```

We will use the joint reducedDims - "LSI_Combined" and joint embeddings - "UMAP_Combined"

```
proj@reducedDims
#> List of length 3
#> names(3): LSI_ATAC LSI_RNA LSI_Combined
proj@embeddings
#> List of length 3
#> names(3): UMAP_Combined UMAP_RNA UMAP_ATAC
```

Retrieve gene expression matrix from the ArchR project

```
GeneExpressionMatrix= getMatrixFromProject(
    ArchRProj = proj,
    useMatrix = "GeneExpressionMatrix",
    useSeqnames = NULL,
    verbose = TRUE,
    binarize = FALSE,
    threads = getArchRThreads(),
    logFile = createLogFile("getMatrixFromProject")
)
#> ArchR logging to : ArchRLogs/ArchR-getMatrixFromProject-4fa629a8d347-Date-2022-04-21_Time-13-15-51.l
#> If there is an issue, please report to github with logFile!
#> 2022-04-21 13:16:00 : Organizing colData, 0.146 mins elapsed.
#> 2022-04-21 13:16:00 : Organizing rowData, 0.146 mins elapsed.
#> 2022-04-21 13:16:00 : Organizing rowRanges, 0.146 mins elapsed.
#> 2022-04-21 13:16:00 : Organizing Assays (1 of 1), 0.146 mins elapsed.
#> 2022-04-21 13:16:00 : Constructing SummarizedExperiment, 0.147 mins elapsed.
#> 2022-04-21 13:16:00 : Finished Matrix Creation, 0.158 mins elapsed.
```

Change to gene expression matrix to SingleCellExperiment object

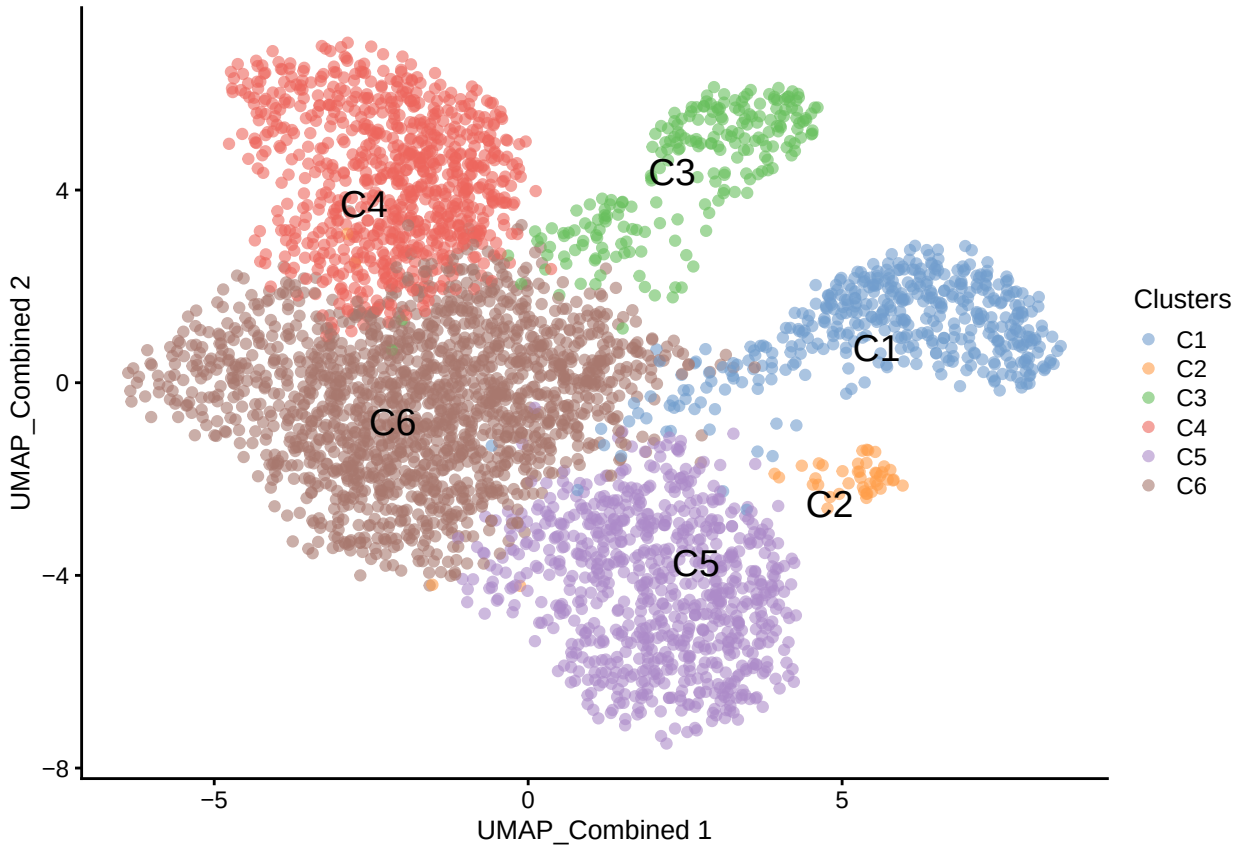
```
GeneExpressionMatrix=as(GeneExpressionMatrix, "SingleCellExperiment")
assay(GeneExpressionMatrix, "logcounts") = assay(GeneExpressionMatrix, "GeneExpressionMatrix")
```

Transfer cell and gene information and embeddings from ArchR project to singleCellExperiment

```
reducedDim(GeneExpressionMatrix, "UMAP_Combined") <- getEmbedding(ArchRProj = proj,
                                                                  embedding = "UMAP_Combined",
                                                                  returnDF =TRUE)[colnames(GeneExpressi
colData(GeneExpressionMatrix)=getCellColData(proj)[colnames(GeneExpressionMatrix),]
rownames(GeneExpressionMatrix)=rowData(GeneExpressionMatrix)$name
```

Visualize singleCellExperiment by UMAP

```
scater::plotReducedDim(GeneExpressionMatrix, dimred = "UMAP_Combined",
                       text_by = "Clusters", colour_by = "Clusters")
```



Quick start

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 mm10 are available.

```
grl <- getTFMotifInfo(genome="hg38")
head(grl)
#> GRangesList object of length 6:
#> $ADNP
#> GRanges object with 20545 ranges and 0 metadata columns:
#>            seqnames              ranges strand
#>               <Rle>           <IRanges>  <Rle>
#>       [1]     chr1       629819-630076      *
#>       [2]     chr1       633892-634164      *
#>       [3]     chr1       960443-960765      *
#>       [4]     chr1     1011312-1012144      *
#>       [5]     chr1     1058025-1058347      *
#>       ...      ...                 ...    ...
#>   [20541]     chrX 154136551-154137792      *
#>   [20542]     chrX 154501525-154502608      *
```

```
#>   [20543]     chrX 154751612-154752444      *
#>   [20544]     chrX 155069184-155070016      *
#>   [20545]     chrX 155218440-155219571      *
#>   -------
#>   seqinfo: 25 sequences from an unspecified genome; no seqlengths
#>
#> ...
#> <5 more elements>
```

2. Link ATAC-seq peaks to target genes

Next, we compute peak to gene correlations using the addPeak2GeneLinks function from the ArchR package. The user would need to supply a path to an ArchR project already containing peak and gene matrices, as well as Latent semantic indexing (LSI) dimensionality reduction.

```
# path to ArchR project
p2g <- calculateP2G(ArchR_path = myarchR_project_path, useDim = "LSI_Combined",
                    useMatrix = "GeneExpressionMatrix")
#> Setting ArchRLogging = FALSE
#> 2022-04-21 13:16:57 : Getting Available Matrices, 0 mins elapsed.
#> No predictionScore found. Continuing without predictionScore!
#> 2022-04-21 13:16:57 : Filtered Low Prediction Score Cells (0 of 3903, 0), 0 mins elapsed.
#> 2022-04-21 13:16:57 : Computing KNN, 0.001 mins elapsed.
#> 2022-04-21 13:17:10 : Identifying Non-Overlapping KNN pairs, 0.212 mins elapsed.
#> 2022-04-21 13:17:13 : Identified 491 Groupings!, 0.261 mins elapsed.
#> 2022-04-21 13:17:13 : Getting Group RNA Matrix, 0.265 mins elapsed.
#> 2022-04-21 13:17:47 : Getting Group ATAC Matrix, 0.83 mins elapsed.
#> 2022-04-21 13:18:41 : Normalizing Group Matrices, 1.731 mins elapsed.
#> 2022-04-21 13:18:46 : Finding Peak Gene Pairings, 1.811 mins elapsed.
#> 2022-04-21 13:18:46 : Computing Correlations, 1.821 mins elapsed.
#> 2022-04-21 13:18:53 : Completed Peak2Gene Correlations!, 1.926 mins elapsed.
head(p2g)
#>   idxATAC Chrom idxRNA       Gene Correlation
#> 1     268  chr1    100        SKI   0.5467930
#> 2     484  chr1    171      RPL22   0.5035875
#> 3     499  chr1    181       ESPN   0.5192811
#> 4     513  chr1    181       ESPN   0.5149298
#> 5     627  chr1    207       UTS2   0.5067324
#> 6     628  chr1    200 AL359881.3   0.8701104
```

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 that has similar performance to ArchR's P2G function.

```
# load the MAE object
mae <- readRDS("/gstore/project/archr_importer/ne_multiome/mae.rds")
# peak matrix
peakmatrix <- mae[["PeakMatrix"]]
# expression matrix
expmatrix <- mae[["GeneIntegrationMatrix"]]
rownames(expmatrix) <- rowData(expmatrix)$name
# dimensional reduction matrix
reducedDim <- SingleCellExperiment::reducedDims(mae[['TileMatrix500']])[["IterativeLSI"]]

p2g <- calculateP2G(peakmatrix, expmatrix, reducedDim)
```

```
head(p2g)
```

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(archR_project_path=myarchR_project_path, grl=grl, p2g=p2g)
#> Successfully loaded ArchRProject!
#> Computing overlap...
#> Success!
```

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=TRUE)
head(regulon)
#>       tf target      corr
#> 1     AR   AAK1 0.5284217
#> 2 ARID1B   AAK1 0.5284217
#> 3   ATF1   AAK1 0.5284217
#> 4   ATF3   AAK1 0.5284217
#> 5  ATOH8   AAK1 0.5284217
#> 6   BCL3   AAK1 0.5284217
```

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.

```
regulon.w <- addWeights(regulon=regulon,
                        sce=GeneExpressionMatrix,
                        cluster_factor="Clusters",
                        exprs_values="GeneExpressionMatrix",
                        block_factor=NULL,
                        corr=TRUE,
                        MI=FALSE,
                        BPPARAM=BiocParallel::MulticoreParam())
#> calculating average expression across clusters...
#> computing correlation of the regulon...
#> if the standard deviation for TF expression is zero, the derived weight will be NA...
#>   |                                                                              |
```

```
#> 8178  ADNP AC010333.1 0.6267067 0.44618082
#> 8762  ADNP AC010333.2 0.6699047 0.23497260
#> 9534  ADNP AC010618.2 0.7700654 0.46512982
#> 11081 ADNP AC012170.2 0.5634615 0.06406767
```

5. 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(GeneExpressionMatrix, regulon.w, "weight",
                                   method="weightedMean", assay="GeneExpressionMatrix")
#> calculating TF activity from regulon using weightedmean
#>   |                                                                              |
```

```
markers <- findDifferentialActivity(score.combine, GeneExpressionMatrix$Clusters,
                                    pval.type="some", direction="up", test.type= "t")
```

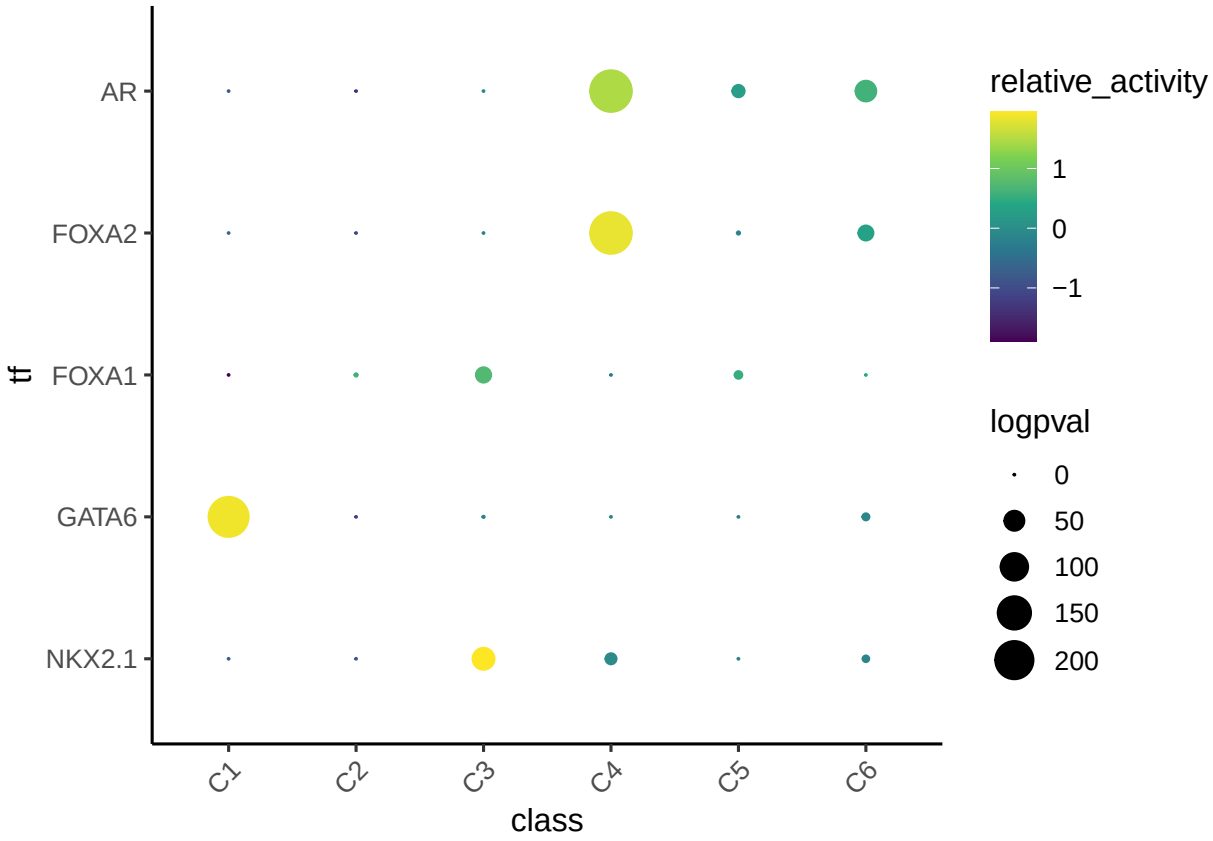
Take the top TFs

```
markers.sig <- getSigGenes(markers, topgenes = 5 )
#> Using a logFC cutoff of 1 for class 1
#> Using a logFC cutoff of 1.1 for class 2
#> Using a logFC cutoff of 0.3 for class 3
#> Using a logFC cutoff of 1 for class 4
#> Using a logFC cutoff of 0.4 for class 5
#> Using a logFC cutoff of 0.6 for class 6
```

7. Visualize the results

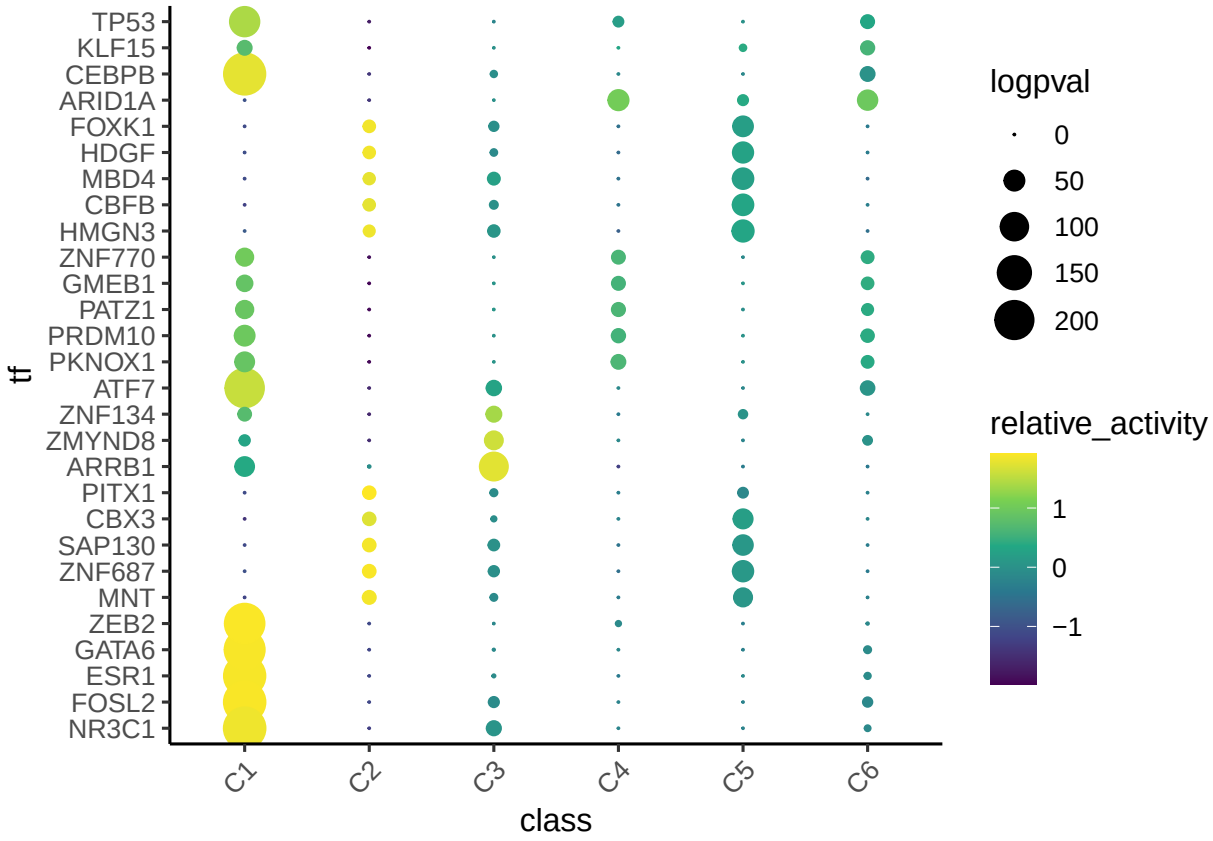
First visualize the known differential TFs by bubble plot

```
plotBubble(activity_matrix = score.combine, tf = c("NKX2-1","GATA6","FOXA1","FOXA2", "AR"), class=GeneE
```



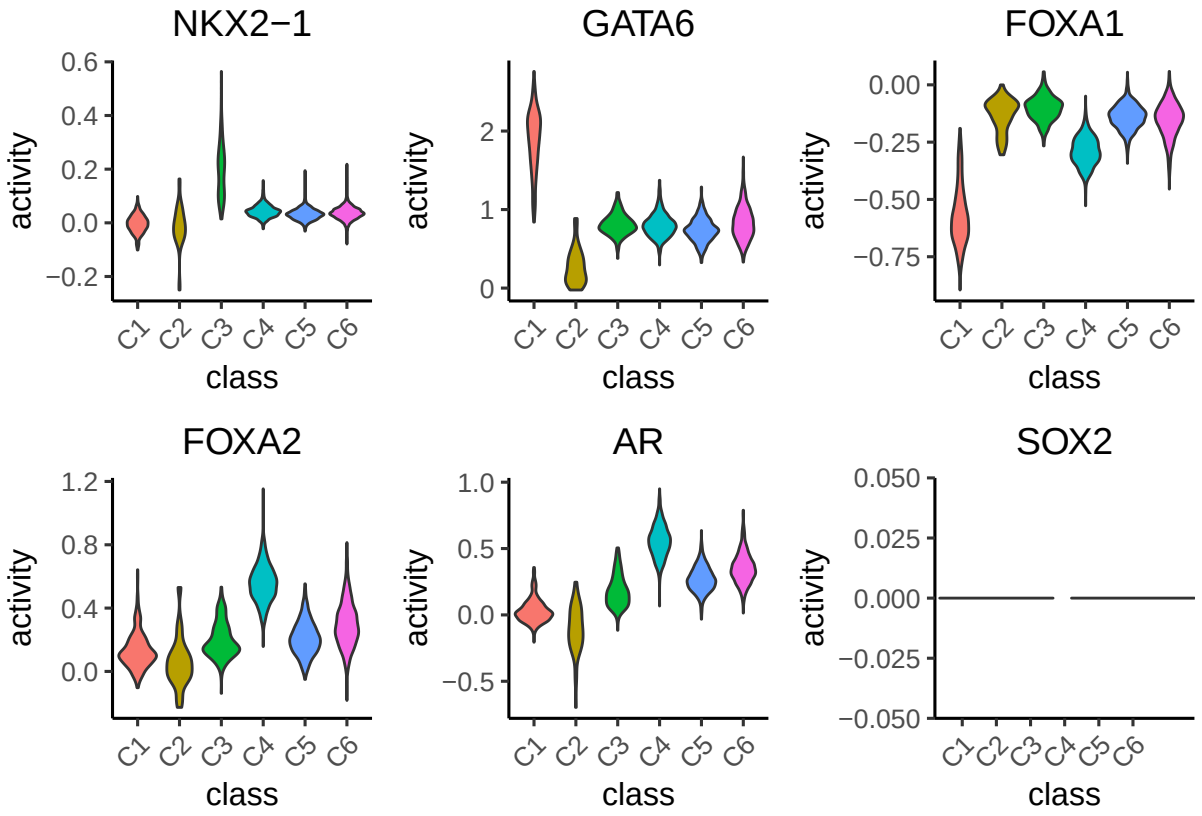
Then visualize the most differential TFs by clusters

```
plotBubble(activity_matrix = score.combine, tf = markers.sig$tf,
           class=GeneExpressionMatrix$Clusters)
```



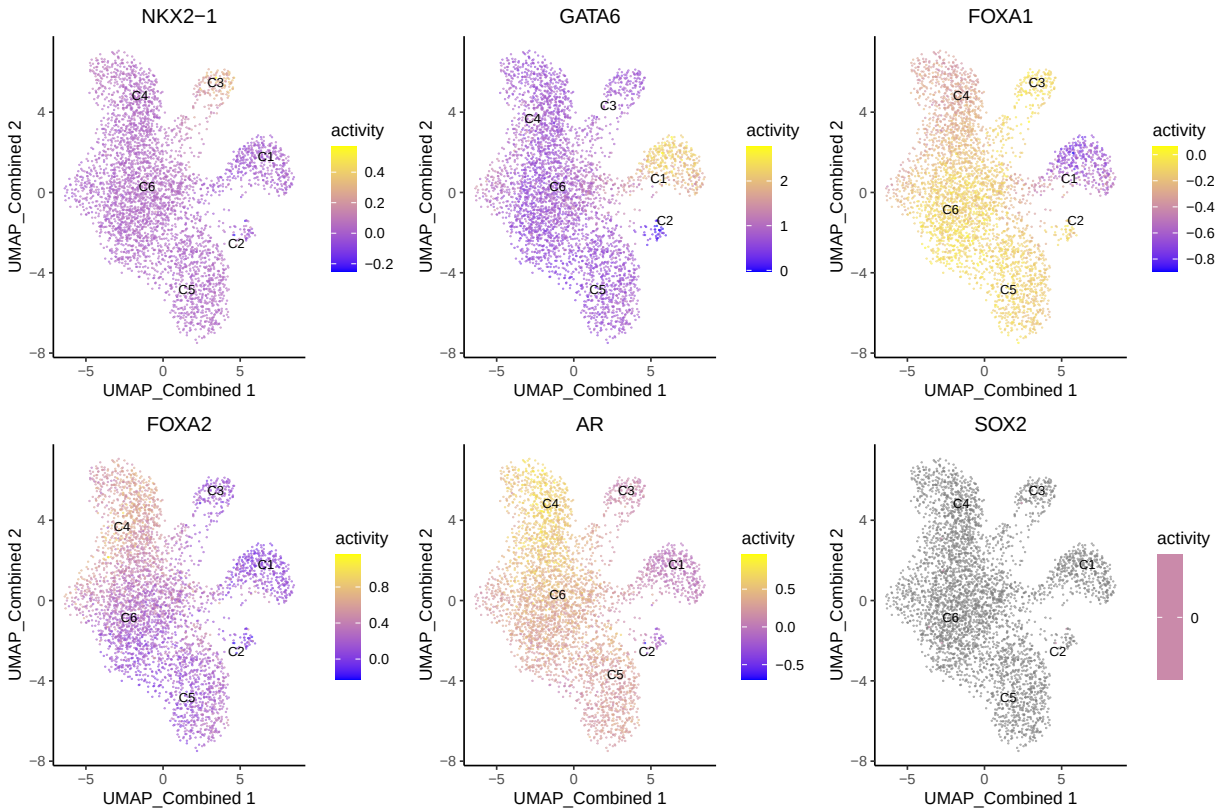
Visualize the known differential TFs by violin plot. Note there is no activity calculated for SOX2 because the expression of SOX2 is 0 in all cells.

```
plotActivityViolin(score.combine, c("NKX2-1","GATA6","FOXA1","FOXA2", "AR", "SOX2"),
                   GeneExpressionMatrix$Clusters)
#> Warning: Removed 3896 rows containing non-finite values (stat_ydensity).
#> Warning: Groups with fewer than two data points have been dropped.
#> Groups with fewer than two data points have been dropped.
```



Visualize the known differential TFs by UMAP

```
plotActivityDim(sce=GeneExpressionMatrix, activity_matrix=score.combine,
                tf=c("NKX2-1","GATA6","FOXA1","FOXA2", "AR", "SOX2"),
                dimtype="UMAP_Combined", label = "Clusters", point_size=0.1, ncol = 3)
```



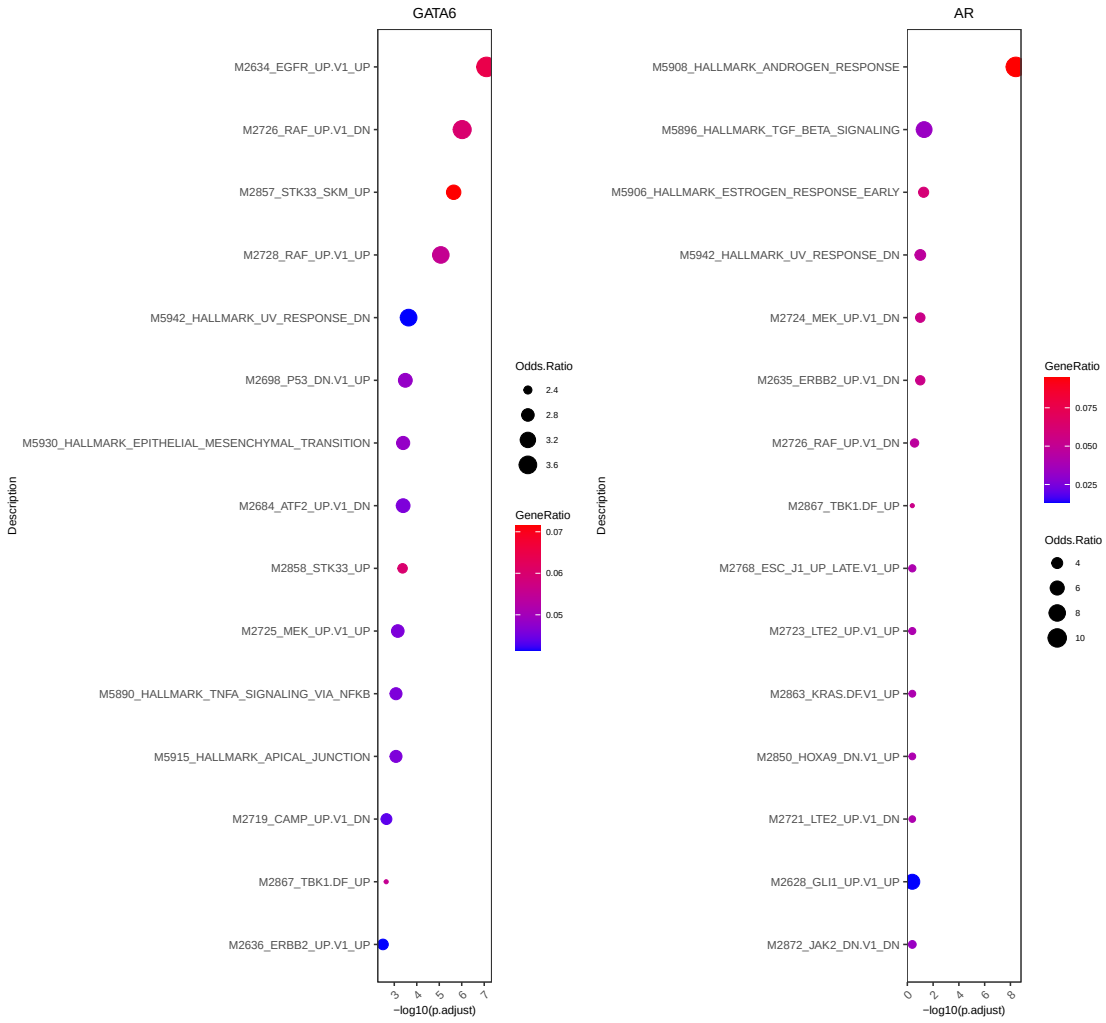
8. Geneset enrichment Sometimes we are interested to know what pathways are enriched in the regulon of a particular TF. We can perform geneset enrichment using the enricher function from clusterProfiler.

```
#retrieve genesets
H = EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb", cat = "H", gene.id.type = "SYMBOL" )
C6 = EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb", cat = "C6", gene.id.type = "SYMBOL" )

#combine genesets and convert genesets to be compatible with enricher
gs=c(H,C6)
gs.list=do.call(rbind,lapply(names(gs), function(x) {data.frame(gs=x, genes=gs[[x]])}))

enrichresults=regulonEnrich(c("GATA6","AR"), regulon=regulon.w, corr="weight",
                            corr_cutoff=0.5, genesets=gs.list)
#>

#plot results
enrichPlot(results=enrichresults)
#> Warning: `panel.margin` is deprecated. Please use `panel.spacing` property instead
#> `panel.margin` is deprecated. Please use `panel.spacing` property instead
```



Session Info

```
sessionInfo()
#> R Under development (unstable) (2022-02-23 r81801)
#> 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         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
#>
#> attached base packages:
```

```
#> [1] parallel  stats4    stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] org.Hs.eg.db_3.15.0         AnnotationDbi_1.57.1
#>  [3] msigdbr_7.5.1               nabor_0.5.0
#>  [5] epiregulon_1.0.2            SingleCellExperiment_1.17.2
#>  [7] ArchR_1.0.1                 magrittr_2.0.3
#>  [9] rhdf5_2.39.6                Matrix_1.4-0
#> [11] data.table_1.14.2           SummarizedExperiment_1.25.3
#> [13] Biobase_2.55.2              GenomicRanges_1.47.6
#> [15] GenomeInfoDb_1.31.7         IRanges_2.29.1
#> [17] S4Vectors_0.33.17           BiocGenerics_0.41.2
#> [19] MatrixGenerics_1.7.0        matrixStats_0.62.0
#> [21] ggplot2_3.3.5
#>
#> loaded via a namespace (and not attached):
#>   [1] utf8_1.2.2                tidyselect_1.1.2
#>   [3] RSQLite_2.2.12            grid_4.2.0
#>   [5] BiocParallel_1.29.21      scatterpie_0.1.7
#>   [7] munsell_0.5.0             ScaledMatrix_1.3.0
#>   [9] base64url_1.4             codetools_0.2-18
#>  [11] statmod_1.4.36            scran_1.23.1
#>  [13] withr_2.5.0               colorspace_2.0-3
#>  [15] GOSemSim_2.21.1           genomitory_1.99.3
#>  [17] filelock_1.0.2            highr_0.9
#>  [19] knitr_1.38                rstudioapi_0.13
#>  [21] DOSE_3.21.2               labeling_0.4.2
#>  [23] KEGGgraph_1.55.0          GenomeInfoDbData_1.2.8
#>  [25] polyclip_1.10-0           bit64_4.0.5
#>  [27] farver_2.1.0              downloader_0.4
#>  [29] treeio_1.19.2             vctrs_0.3.8
#>  [31] generics_0.1.2            xfun_0.30
#>  [33] BiocFileCache_2.3.4       R6_2.5.1
#>  [35] ggbeeswarm_0.6.0          graphlayouts_0.8.0
#>  [37] rsvd_1.0.5                gp.version_1.5.0
#>  [39] locfit_1.5-9.5            bitops_1.0-7
#>  [41] rhdf5filters_1.7.0        cachem_1.0.6
#>  [43] fgsea_1.21.2              gridGraphics_0.5-1
#>  [45] DelayedArray_0.21.2       assertthat_0.2.1
#>  [47] scales_1.2.0              ggraph_2.0.5
#>  [49] enrichplot_1.15.3         beeswarm_0.4.0
#>  [51] gtable_0.3.0              beachmat_2.11.0
#>  [53] metacommons_1.7.2         tidygraph_1.2.1
#>  [55] rlang_1.0.2               splines_4.2.0
#>  [57] lazyeval_0.2.2            gp.auth_1.5.2
#>  [59] yaml_2.3.5                reshape2_1.4.4
#>  [61] backports_1.4.1           qvalue_2.27.0
#>  [63] clusterProfiler_4.3.4     tools_4.2.0
#>  [65] ggplotify_0.1.0           ellipsis_0.3.2
#>  [67] RColorBrewer_1.1-3        artificer.base_1.1.5
#>  [69] Rcpp_1.0.8.3              plyr_1.8.7
#>  [71] sparseMatrixStats_1.7.0   zlibbioc_1.41.0
```

```
#>  [73] purrr_0.3.4                RCurl_1.98-1.6
#>  [75] artificer.ranges_1.1.0     viridis_0.6.2
#>  [77] cowplot_1.1.1              ggrepel_0.9.1
#>  [79] cluster_2.1.2              DO.db_2.9
#>  [81] patchwork_1.1.1            evaluate_0.15
#>  [83] xtable_1.8-4               XML_3.99-0.9
#>  [85] gridExtra_2.3              compiler_4.2.0
#>  [87] scater_1.23.6              tibble_3.1.6
#>  [89] shadowtext_0.1.1           crayon_1.5.0
#>  [91] gp.cache_1.5.4             htmltools_0.5.2
#>  [93] ggfun_0.0.6                ArtifactDB_1.7.4
#>  [95] tidyr_1.2.0                aplot_0.1.3
#>  [97] DBI_1.1.2                  tweenr_1.0.2
#>  [99] dbplyr_2.1.1               MASS_7.3-55
#> [101] rappdirs_0.3.3             babelgene_22.3
#> [103] cli_3.2.0                  metapod_1.3.0
#> [105] igraph_1.3.1               pkgconfig_2.0.3
#> [107] getPass_0.2-2              scuttle_1.5.2
#> [109] ggtree_3.3.2               annotate_1.73.0
#> [111] vipor_0.4.5                dqrng_0.3.0
#> [113] XVector_0.35.0             yulab.utils_0.0.4
#> [115] stringr_1.4.0              digest_0.6.29
#> [117] graph_1.73.0               Biostrings_2.63.3
#> [119] rmarkdown_2.13             fastmatch_1.1-3
#> [121] tidytree_0.3.9             edgeR_3.37.1
#> [123] DelayedMatrixStats_1.17.0 GSEABase_1.57.0
#> [125] curl_4.3.2                 gtools_3.9.2
#> [127] nlme_3.1-157               lifecycle_1.0.1
#> [129] jsonlite_1.8.0             Rhdf5lib_1.17.3
#> [131] BiocNeighbors_1.13.0       viridisLite_0.4.0
#> [133] limma_3.51.8               fansi_1.0.3
#> [135] pillar_1.7.0               lattice_0.20-45
#> [137] KEGGREST_1.35.0            fastmap_1.1.0
#> [139] httr_1.4.2                 GO.db_3.15.0
#> [141] glue_1.6.2                 png_0.1-7
#> [143] bluster_1.5.1              bit_4.0.4
#> [145] Rgraphviz_2.39.1           ggforce_0.3.3
#> [147] stringi_1.7.6              EnrichmentBrowser_2.25.3
#> [149] blob_1.2.3                 BiocSingular_1.11.0
#> [151] memoise_2.0.1              dplyr_1.0.8
#> [153] irlba_2.3.5                ape_5.6-2
```