

# Multiome tutorial - MultiAssayExperiment

**Xiaosai Yao**

**27 November 2022**

## Package

epiregulon 1.0.19

## Contents

1	Introduction	2
2	Installation	2
3	Data preparation	2
4	Quick start	3
4.1	Retrieve bulk TF ChIP-seq binding sites	3
4.2	Link ATAC-seq peaks to target genes	4
4.3	Add TF motif binding to peaks	4
4.4	Generate regulons	5
4.5	Network pruning (highly recommended)	5
4.6	Add Weights	5
4.7	Calculate TF activity	6
4.8	Perform differential activity	7
4.9	Visualize the results	7
4.10	Geneset enrichment	10
4.11	Network analysis	11
4.12	Differential networks	11
5	Session Info	14

## 1 Introduction

---

This tutorial walks through the same dataset used in the “multiome tutorial - archR workflow”. This is a dataset generated by infecting LNCaP cells with NKX2-1 and GATA6 to examine the effects of these TFs on AR activity.

## 2 Installation

---

Epiregulon is currently available on R/dev

```
library(epiregulon)
```

Alternatively, you could install from gitlab

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

library(epiregulon)
```

## 3 Data preparation

---

Single cell preprocessing needs to be performed by user’s favorite methods prior to using Epiregulon. The following components are required: 1. Peak matrix from scATAC-seq 2. Gene expression matrix from either paired or unpaired scRNA-seq. RNA-seq integration needs to be performed for unpaired dataset. 3. Dimensionality reduction matrix from with either single modalities or joint scRNA-seq and scATAC-seq

Multiome data can now be conveniently processed by `initiate.archr` and then `gp.sa.archr` to obtain peak matrices. Finally, the archR project can be uploaded into DatasetDB as a `MultiAssayExperiment` object using `maw.archr::importArchr` or `maw.archr::create.mae.with.multiple.sces.from.archr`

```
# load the MAE object
library(SingleCellExperiment)
mae <- dsassembly::getDataset("DS000013080")
#> 'version=' not specified, using the latest version (2) instead
#> Error in read_csv(path, is_compressed = identical(compression, "gzip"), :
#>   encountered empty line in a file with non-zero columns
#>   falling back to 'read.csv' for a malformed CSV data frame

# peak matrix
PeakMatrix <- mae[["PeakMatrix"]]

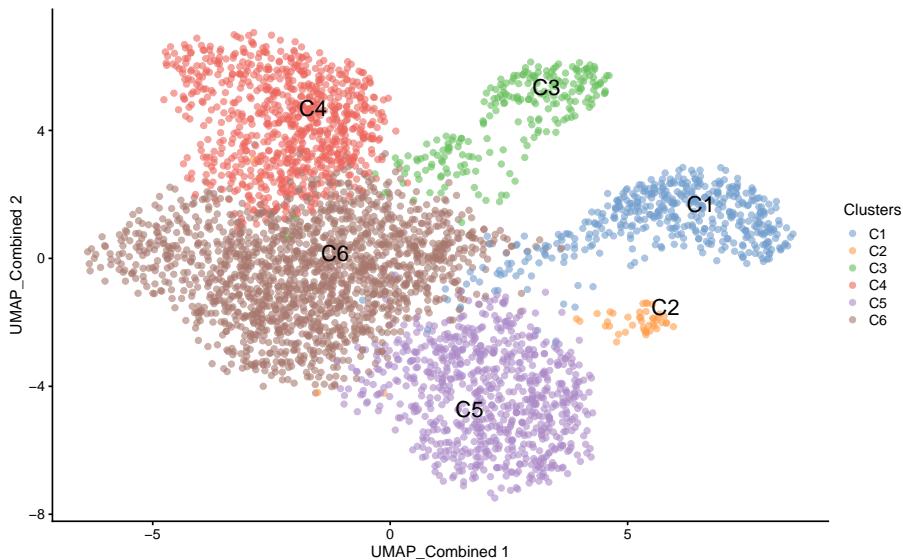
# expression matrix
GeneExpressionMatrix <- mae[["GeneExpressionMatrix"]]
rownames(GeneExpressionMatrix) <- rowData(GeneExpressionMatrix)$name

# dimensional reduction matrix
reducedDimMatrix <- reducedDim(mae[['TileMatrix500']], "LSI_ATAC")
```

## Multiome tutorial - MultiAssayExperiment

Visualize singleCellExperiment by UMAP

```
# transfer UMAP_combined from TileMatrix to GeneExpressionMatrix
reducedDim(GeneExpressionMatrix, "UMAP_Combined") <- reducedDim(mae[['TileMatrix500']], "UMAP_Combined")
scater::plotReducedDim(GeneExpressionMatrix,
                       dimred = "UMAP_Combined",
                       text_by = "Clusters",
                       colour_by = "Clusters")
```



## 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 mm10 are available.

```
grl <- getTFMotifInfo(genome = "hg38")
#> redirecting from 'GMTY162:hg38_motif_bed_granges@REVISION-4' to 'GMTY162:hg38_motif_bed_granges@24c22e4f4'
head(grl)
#> GRangesList object of length 6:
#> $`5-hmC`
#> GRanges object with 24048 ranges and 0 metadata columns:
#>   seqnames      ranges strand
#>   <Rle>      <IRanges>  <Rle>
#>   [1] chr1    10000-10685    *
#>   [2] chr1    13362-13694    *
#>   [3] chr1    29631-29989    *
#>   [4] chr1    40454-40754    *
#>   [5] chr1    135395-135871   *
```

## Multiome tutorial - MultiAssayExperiment

```
#> [24044] chrY 56864377-56864627      *
#> [24045] chrY 56876124-56876182      *
#> [24046] chrM     84-2450          *
#> [24047] chrM     13613-14955        *
#> [24048] chrM     15134-16490        *
#> -----
#> seqinfo: 25 sequences from an unspecified genome; no seqlengths
#>
#> ...
#> <5 more elements>
```

## 4.2 Link ATAC-seq peaks to target genes

Next, we compute peak to gene correlations using a custom algorithm that has similar performance to ArchR's P2G function.

```
p2g <- calculateP2G(peakMatrix = PeakMatrix,
                     expMatrix = GeneExpressionMatrix,
                     reducedDim = reducedDimMatrix)
#> Using epiregulon to compute peak to gene links...
#> performing k means clustering to form metacells
#> Computing correlation

head(p2g)
#>   idxATAC chr start    end idxRNA      target Correlation distance      FDR
#> 67       1 chr1 817121 817621      19  FAM41C  0.5354880  50578 0.33397844
#> 33       5 chr1 858735 859235      17 LINC01128  0.5791498  30938 0.23728337
#> 478      9 chr1 920452 920952      29 AL645608.7 0.5553784  75097 0.28919700
#> 76      10 chr1 920987 921487      19  FAM41C  0.7203272  50587 0.03788523
#> 217      11 chr1 923974 924474      23 LINC02593  0.5332530  4083 0.33907862
#> 257      11 chr1 923974 924474      24  SAMD11  0.5283517      0 0.35007472
```

## 4.3 Add TF motif binding to peaks

The next step is to add the TF binding information by overlapping 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(grl = grl, p2g = p2g, peakMatrix = PeakMatrix)
#> Computing overlap...
#> Success!
head(overlap)
#>   idxATAC idxTF      tf
#> 1       1    2  5-mC
#> 2       1   22 AML1-ETO
#> 3       1   25      AR
```

```
#> 4      1    49    ATF1
#> 5      1    50    ATF2
#> 6      1    51    ATF3
```

## 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 = p2g, overlap = overlap, aggregate = FALSE)
head(regulon)
#>   idxATAC idxTF      tf chr start     end idxRNA target      corr distance      FDR
#> 1      1     2 5-mC chr1 817121 817621     19 FAM41C 0.535488 50578 0.3339784
#> 2      1    22 AML1-ETO chr1 817121 817621     19 FAM41C 0.535488 50578 0.3339784
#> 3      1    25      AR chr1 817121 817621     19 FAM41C 0.535488 50578 0.3339784
#> 4      1    49    ATF1 chr1 817121 817621     19 FAM41C 0.535488 50578 0.3339784
#> 5      1    50    ATF2 chr1 817121 817621     19 FAM41C 0.535488 50578 0.3339784
#> 6      1    51    ATF3 chr1 817121 817621     19 FAM41C 0.535488 50578 0.3339784
```

## 4.5 Network pruning (highly recommended)

Epiregulon prunes the network by performing tests of independence on the observed number of cells jointly expressing transcription factor (TF), regulatory element (RE) and target gene (TG) vs the expected number of cells if TF/RE and TG are independently expressed. We implement two tests, the binomial test and the chi-square test. In the binomial test, the expected probability is  $P(\text{TF}, \text{RE}) * P(\text{TG})$ , and the number of trials is the total number of cells, and the observed successes is the number of cells jointly expressing all three elements. In the chi-square test, the expected probability for having all 3 elements active is also  $P(\text{TF}, \text{RE}) * P(\text{TG})$  and the probability otherwise is  $1 - P(\text{TF}, \text{RE}) * P(\text{TG})$ . The observed cell count for the active category is the number of cells jointly expressing all three elements, and the cell count for the inactive category is  $n - n_{\text{triple}}$ .

We calculate cluster-specific p-values if users supply cluster labels. This is useful if we are interested in cluster-specific networks. The pruned regulons can then be used to visualize differential networks for transcription factors of interest. See section on differential networks.

## 4.6 Add Weights

While the ‘pruneRegulon’ function provides statistics on the joint occurrence of TF-RE-TG, we would like to further estimate the strength of regulation. Biologically, this can be interpreted as the magnitude of gene expression changes induced by transcription factor activity. Epiregulon estimates the regulatory potential using one of the four measures: 1) correlation between TF and target gene expression, 2) mutual information between the TF and target gene expression,

## Multome tutorial - MultiAssayExperiment

3) Wilcoxon test statistics of target gene expression in cells jointly expressing all 3 elements vs cells that do not, or 4) log 2 fold difference of target gene expression in cells jointly expressing all 3 elements vs cells that do not.

Three measures (correlation, Wilcoxon statistics and log 2 fold difference) give both the magnitude and directionality of changes whereas mutational information is always positive. The correlation and mutual information statistics are computed on the grouped pseudobulks by user-supplied cluster labels, whereas the Wilcoxon and log fold change group cells based on the joint expression of TF, RE and TG in each single cell.

```
head(regulon.w)
#>      idxATAC idxTF   tf  chr    start      end idxRNA target      corr distance        FDR      pval_a
#> 2263       22     5 ADNP chr1  960317  960817     28 PERM1 0.6838644  18210 7.050257e-02 1.062648e-0
#> 30325      655     5 ADNP chr1  8061169  8061669    208 TNFRSF9 0.7591704 120104 1.711043e-02 5.790391e-0
#> 33491      787     5 ADNP chr1  9290024  9290524    229 SPSB1 0.9665665  368 4.858063e-08 1.634348e-0
#> 38026      1026     5 ADNP chr1 11663300 11663800    283 AGTRAP 0.5113198  70282 3.888568e-01 6.444799e-0
#> 48759      1259     5 ADNP chr1 16013485 16013985    372 ZBTB17 0.5648523  37207 2.677629e-01 6.491112e-0
#> 48952      1260     5 ADNP chr1 16032798 16033298    379 EPHA2 0.5519491 120769 2.970493e-01 1.714003e-0
#>          pval_C1 pval_C2 pval_C3 pval_C4 pval_C5 pval_C6 stats_all stats_C1 stats_C2 s
#> 2263 0.8357430    NaN 0.8157231 7.803215e-01      NaN 6.149301e-05 2.608910 0.04299061      NaN 0.05
#> 30325 0.1049407    NaN      NaN 6.168540e-19 0.82124525 8.848119e-01 11.842183 2.62880407      NaN
#> 33491 0.4644248    NaN      NaN 8.424454e-01      NaN 9.011359e-01 36.367149 0.53520957      NaN
#> 38026      NaN      NaN 1.831398e-02 0.50841047 4.703493e-01 3.419053      NaN      NaN
#> 48759 0.7417162    NaN      NaN 7.319805e-01 0.01770923 2.914647e-06 11.629583 0.10862397      NaN
#> 48952 0.8587396    NaN 0.9242098 8.400352e-01 0.89231931 6.573484e-08 5.681978 0.03167648      NaN 0.009
#>          stats_C4 stats_C5 stats_C6 padj_all padj_C1 padj_C2 padj_C3      padj_C4 padj_C5 padj_C6
#> 2263 0.07778525    NaN 16.05610618 1.000000000      1      NaN      1 1.000000e+00      NaN 1.000000
#> 30325 79.01354997 0.05105053 0.02098793 1.000000000      1      NaN      NaN 1.998624e-12      1 1.000000
#> 33491 0.03950810    NaN 0.01543225 0.005266795      1      NaN      NaN 1.000000e+00      NaN 1.000000
#> 38026 5.56586302 0.43733723 0.52115526 1.000000000      NaN      NaN      NaN 1.000000e+00      1 1.000000
#> 48759 0.11730094 5.62468684 21.87194146 1.000000000      1      NaN      NaN 1.000000e+00      1 1.000000
#> 48952 0.04074279 0.01832505 29.18652322 1.000000000      1      NaN      1 1.000000e+00      1 0.212114
#>          weight
#> 2263 0.4561281
#> 30325 0.8466490
#> 33491 0.9980748
#> 38026 -0.3203506
#> 48759 -0.6351971
#> 48952 0.1076902
```

## 4.7 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 test statistics of choice, chosen from either correlation, mutual information, Wilcoxon test statistics or log fold change.

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

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

## Multiome tutorial - MultiAssayExperiment

```
score.combine <- calculateActivity(expMatrix = GeneExpressionMatrix,
                                    regulon = regulon.w,
                                    mode = "weight",
                                    method = "weightedMean",
                                    exp_assay = "counts",
                                    normalize = FALSE)
#> calculating TF activity from regulon using weightedmean
```

## 4.8 Perform differential activity

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

Take the top TFs

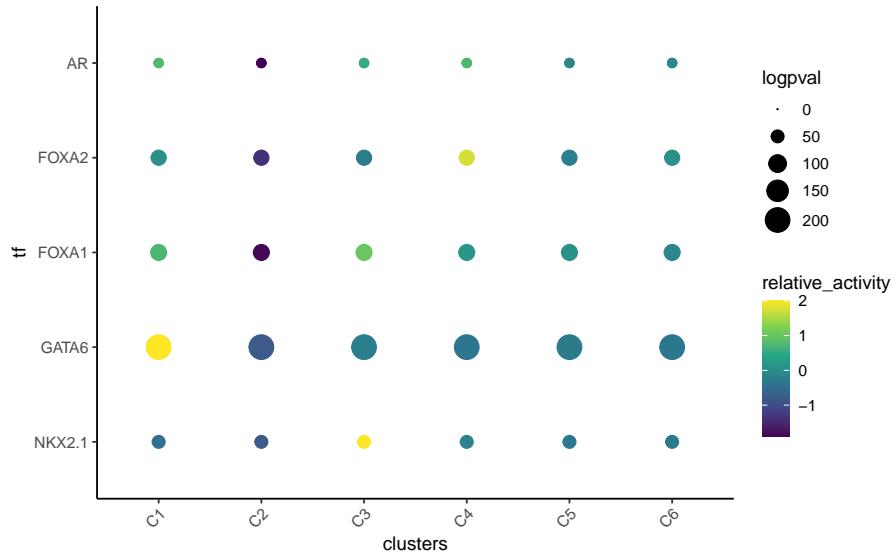
```
markers.sig <- getSigGenes(markers, topgenes = 5 )
#> Using a logFC cutoff of 0.3 for class C1
#> Using a logFC cutoff of 0.1 for class C2
#> Using a logFC cutoff of 0.2 for class C3
#> Using a logFC cutoff of 0.1 for class C4
#> Using a logFC cutoff of 0.2 for class C5
#> Using a logFC cutoff of 0 for class C6
```

## 4.9 Visualize the results

First visualize the known differential TFs by bubble plot

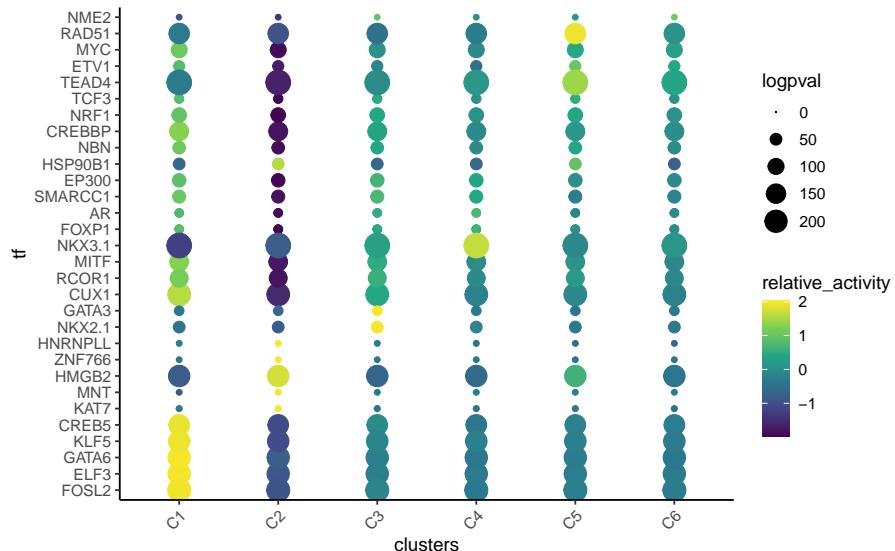
```
plotBubble(activity_matrix = score.combine,
           tf = c("NKX2-1", "GATA6", "FOXA1", "FOXA2", "AR"),
           clusters = GeneExpressionMatrix$Clusters)
```

## Multiome tutorial - MultiAssayExperiment



Then visualize the most differential TFs by clusters

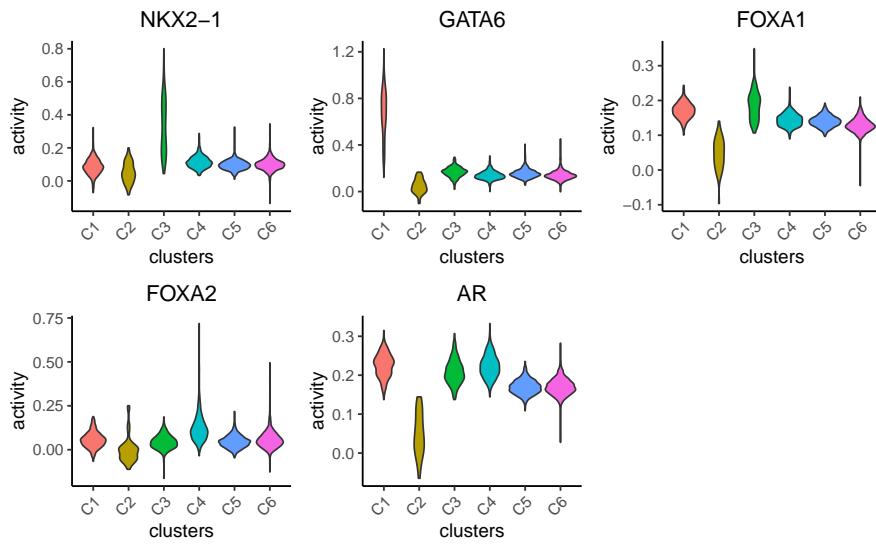
```
plotBubble(activity_matrix = score.combine,
           tf = markers.sig$tf,
           clusters = GeneExpressionMatrix$Clusters)
```



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.

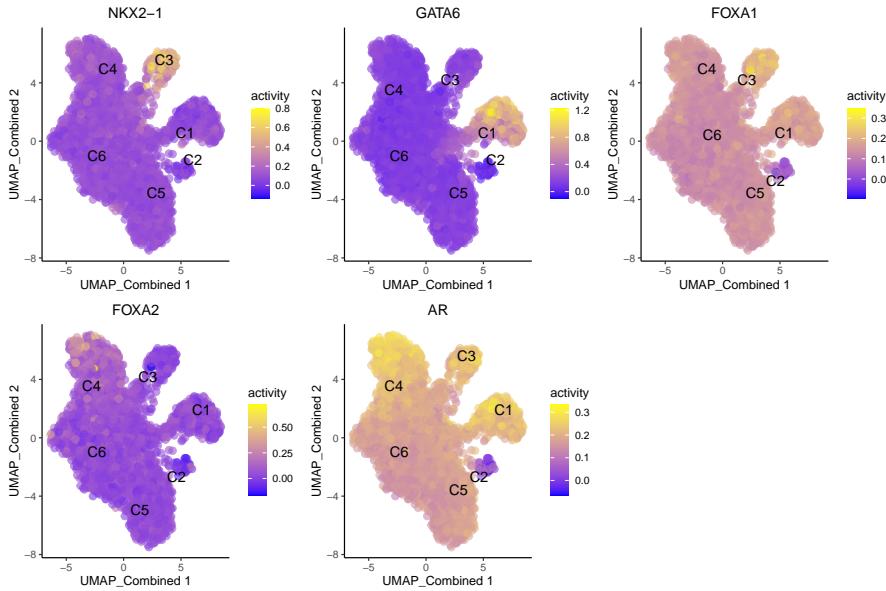
```
plotActivityViolin(activity_matrix = score.combine,
                    tf = c("NKX2-1", "GATA6", "FOXA1", "FOXA2", "AR", "SOX2"),
                    clusters = GeneExpressionMatrix$Clusters)
#> SOX2 not found in activity matrix. Excluded from plots
```

## Multiome tutorial - MultiAssayExperiment



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 = 3,
                 ncol = 3)
#> SOX2 not found in activity matrix. Excluded from plots
```



## 4.10 Geneset enrichment

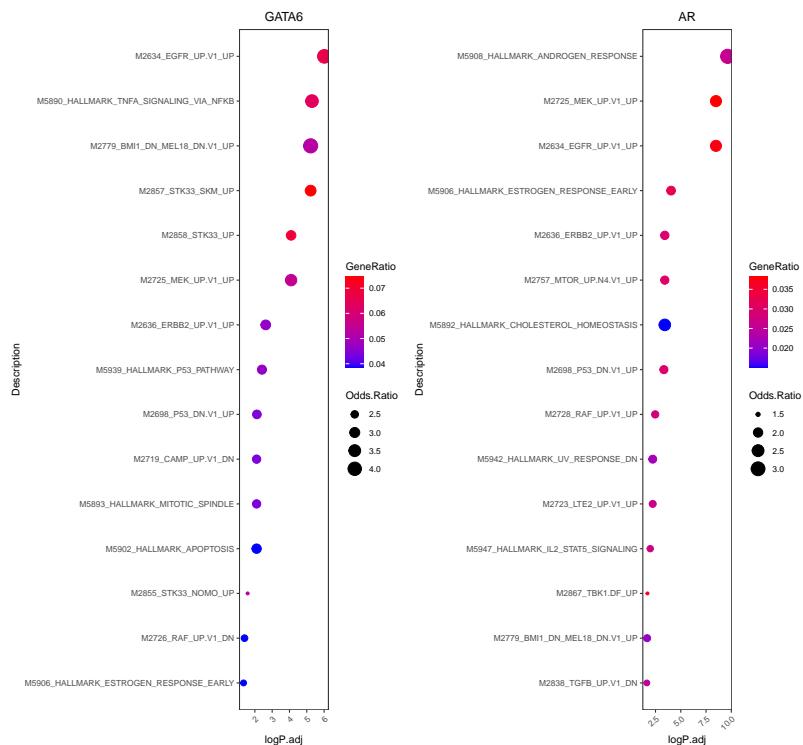
Sometimes we are interested to know what pathways are enriched in the regulon of a particular TF. We can perform geneset enrichment using the enricher function from [clusterProfiler](#).

```
#retrieve genesets
H <- EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb", cat = "H",
                                         gene.id.type = "SYMBOL")
#> Using cached version from 2022-11-23 07:10:49
C6 <- EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb", cat = "C6",
                                         gene.id.type = "SYMBOL")
#> Using cached version from 2022-11-23 07:10:54

#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(TF = c("GATA6","AR"),
                                 regulon = regulon.w,
                                 corr = "weight",
                                 corr_cutoff = 0.1,
                                 genesets = gs.list)

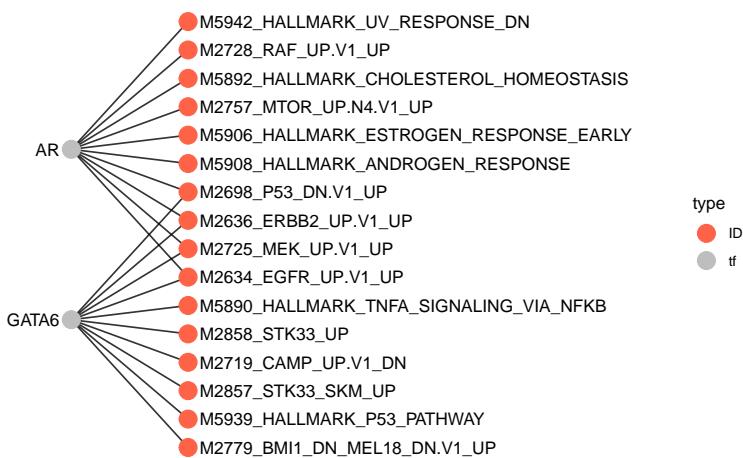
#plot results
enrichPlot(results = enrichresults, )
```



## 4.11 Network analysis

We can visualize the genesets as a network

```
plotGseaNetwork(tf = names(enrichresults),
                enrichresults = enrichresults,
                p.adj_cutoff = 0.1,
                ntop_pathways = 10)
```

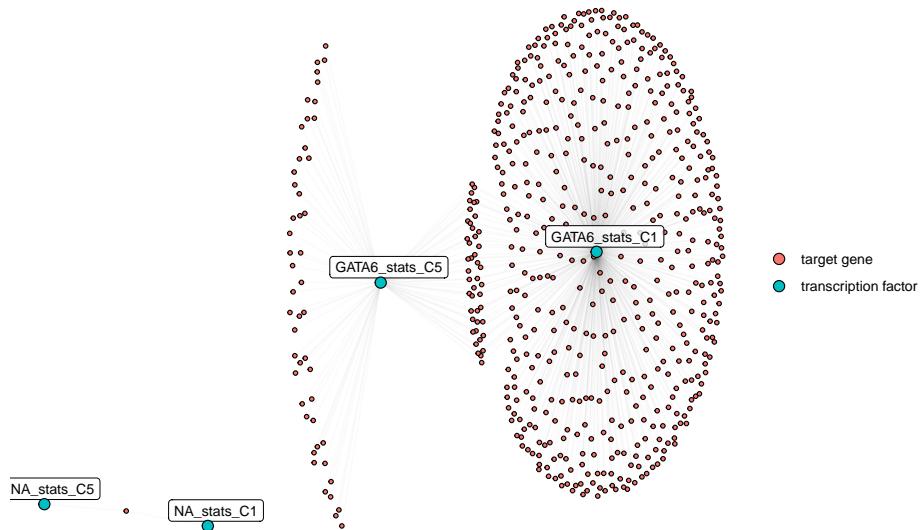


## 4.12 Differential networks

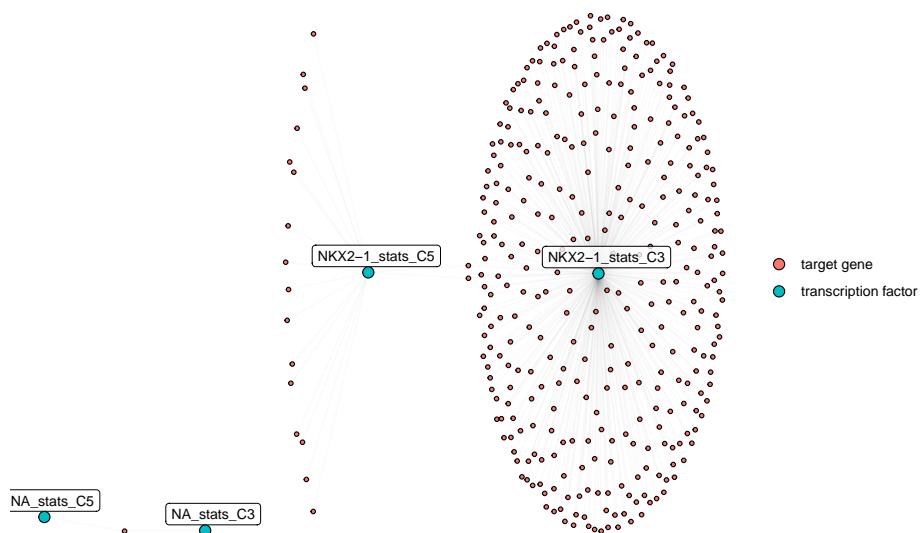
We are interested in understanding the differential networks between two conditions and determining which transcription factors account for the differences in the topology of networks. The pruned regulons with cluster-specific test statistics computed by `pruneRegulon` can be used to generate cluster-specific networks based on user-defined cutoffs and to visualize differential networks for transcription factors of interest. In this dataset, the GATA6 gene was only expressed in cluster 1 (C1) and NKX2-1 was only expressed in cluster 3 (C3). If we visualize the target genes of GATA6, we can see that C1 has many more target genes of GATA6 compared to C5, a cluster that does not express GATA6. Similarly, NKX2-1 target genes are confined to C3 which is the only cluster that exogenously expresses NKX2-1.

```
plotDiffNetwork(pruned.regulon,
                cutoff = 1,
                tf = c("GATA6"),
                groups = c("stats_C1", "stats_C5"),
                layout = "stress")
```

## Multiome tutorial - MultiAssayExperiment



```
plotDiffNetwork(pruned.regulon,
                cutoff = 1,
                tf = c("NKX2-1"),
                groups = c("stats_C3", "stats_C5"),
                layout = "stress")
```



We can also visualize how transcription factors relate to other transcription factors in each cluster.

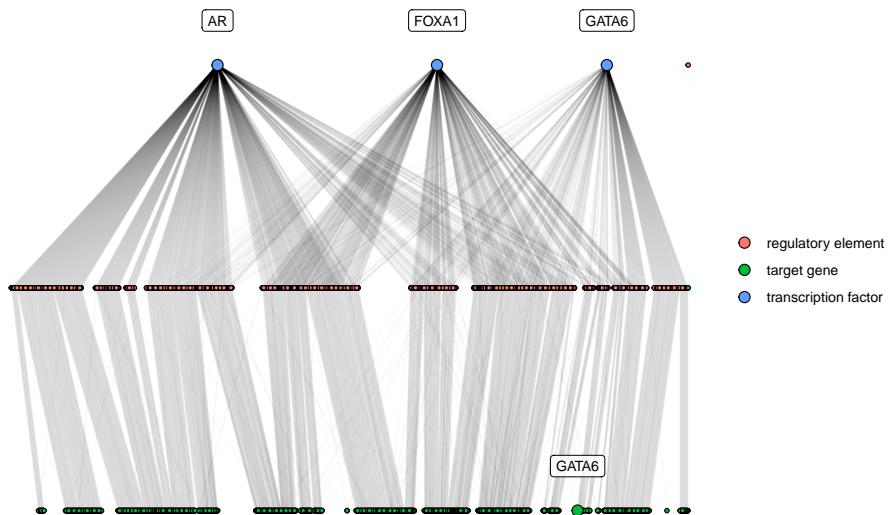
```
C1_network <- buildGraph(pruned.regulon[pruned.regulon$stats_C1>1 & pruned.regulon$tf %in% c("GATA6", "FOXA1")], directed=TRUE)
C5_network <- buildGraph(pruned.regulon[pruned.regulon$stats_C5>1 & pruned.regulon$tf %in% c("GATA6", "FOXA1")], directed=TRUE)

plotEpiRegulonNetwork(C1_network,
                      layout = "sugiyama",
```

## Multiome tutorial - MultiAssayExperiment

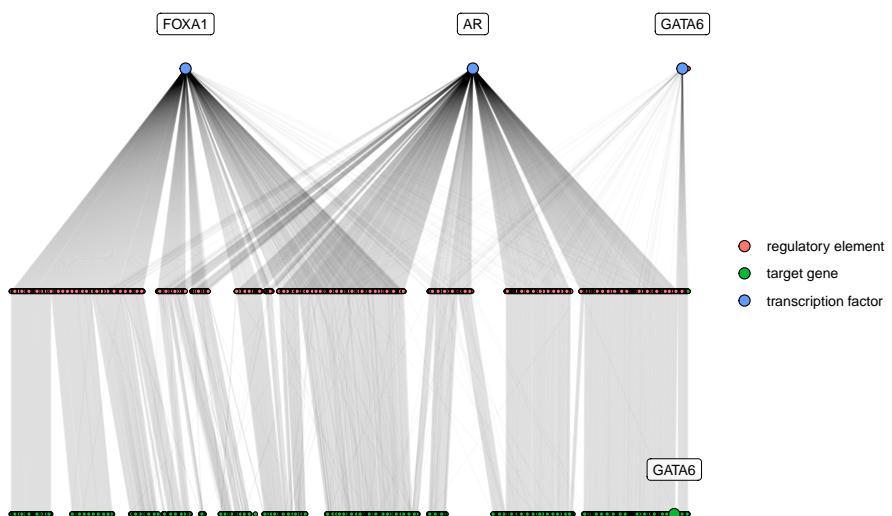
```
tfss_to_highlight = c("GATA6", "FOXA1", "AR")) + ggplot2::ggtitle ("C1")
```

C1



```
plotEpiregulonNetwork(C5_network,
                      layout = "sugiyama",
                      tfss_to_highlight = c("GATA6", "FOXA1", "AR")) + ggplot2::ggtitle ("C5")
```

C5



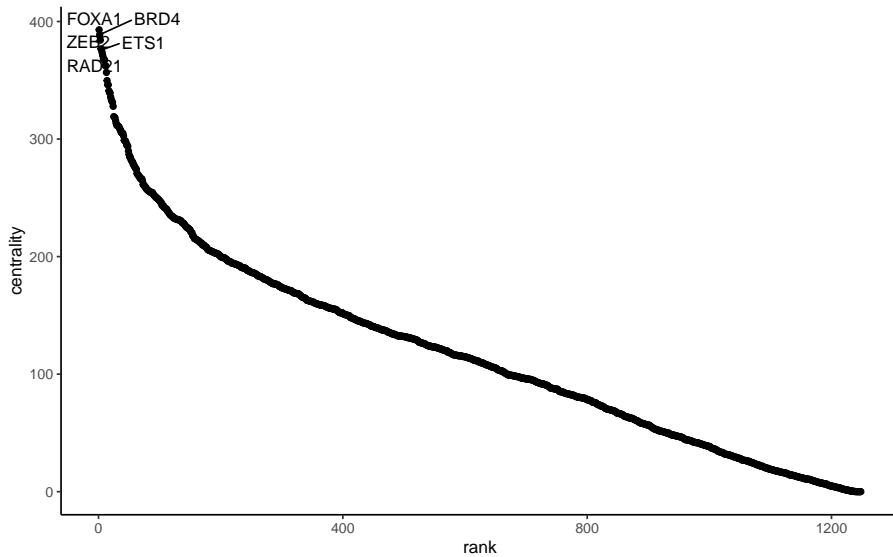
To systematically examine the differential network topology between two clusters, we perform an edge subtraction between two graphs, using weights computed by `pruneRegulon`. We then calculate the degree centrality of the weighted differential graphs and if desired, normalize the differential centrality against the total number of edges. The default normalization function is `sqrt` as it preserves both the difference in the number of edges (but scaled by `sqrt`) and the differences in the weights. If the user only wants to examine the differences in the averaged weights, the `FUN` argument can be changed to `identity`. Finally, we rank the transcription factors by (normalized) differential centrality.

## Multiome tutorial - MultiAssayExperiment

```
# rank by differential centrality
C1_network <- buildGraph(pruned.regulon, weights = "stats_C1")
C5_network <- buildGraph(pruned.regulon, weights = "stats_C5")

diff_graph <- buildDiffGraph(C1_network, C5_network)
diff_graph <- addCentrality(diff_graph)
diff_graph <- normalizeCentrality(diff_graph)
rank_table <- rankTfs(diff_graph)

library(ggplot2)
ggplot(rank_table, aes(x = rank, y = centrality)) +
  geom_point() +
  ggrepel::geom_text_repel(data = head(rank_table, 5), aes(label = tf)) +
  theme_classic()
```



## 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
#>
#> locale:
#> [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C LC_TIME=C LC_COLLATE=C LC_MONETARY=C
#> [6] LC_MESSAGES=C LC_PAPER=C LC_NAME=C LC_ADDRESS=C LC_TELEPHONE=C
```

## Multome tutorial - MultiAssayExperiment

```
#> [11] LC_MEASUREMENT=C      LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] stats4     stats      graphics   grDevices  utils      datasets   methods    base
#>
#> other attached packages:
#> [1] msigdbr_7.5.1           epiregulon_1.0.19        BiocStyle_2.26.0
#> [4] ggplot2_3.4.0            batchtoolsSSH_0.3.6       checkmate_2.1.0
#> [7] MultiAssayExperiment_1.24.0 testthat_3.1.4        SingleCellExperiment_1.20.0
#> [10] SummarizedExperiment_1.29.1 Biobase_2.58.0        GenomicRanges_1.50.1
#> [13] GenomeInfoDb_1.34.3      IRanges_2.32.0          S4Vectors_0.36.0
#> [16] BiocGenerics_0.44.0      MatrixGenerics_1.10.0      matrixStats_0.63.0
#> [19] modeltools_0.2-23
#>
#> loaded via a namespace (and not attached):
#> [1] rappdirs_0.3.3           tidyverse_1.2.1          bit64_4.0.5
#> [5] irlba_2.3.5.1            multcomp_1.4-20         DelayedArray_0.24.0
#> [9] KEGGREST_1.38.0          RCurl_1.98-1.9          generics_0.1.3
#> [13] cowplot_1.1.1           TH.data_1.1-1           ScaledMatrix_1.6.0
#> [17] artificer.mae_1.3.4     bit_4.0.5                shadowtext_0.1.2
#> [21] gp.cache_1.7.1          assertthat_0.2.1        base64url_1.4
#> [25] viridis_0.6.2           xfun_0.31               genomitory_2.1.6
#> [29] babelgene_22.9          evaluate_0.18          jquerylib_0.1.4
#> [33] dbplyr_2.2.1            Rgraphviz_2.42.0        progress_1.2.2
#> [37] purrr_0.3.5             ellipsis_0.3.2          DBI_1.1.3
#> [41] bookdown_0.30            annotate_1.76.0         backports_1.4.1
#> [45] artificer.matrix_1.3.7  vctrs_0.5.1              sparseMatrixStats_1.10
#> [49] withr_2.5.0             ggforce_0.4.1           cachem_1.0.6
#> [53] metacommons_1.9.0        prettyunits_1.1.1        treeio_1.22.0
#> [57] DOSE_3.23.3              BiocBaseUtils_1.1.0      cluster_2.1.3
#> [61] crayon_1.5.1            edgeR_3.40.0             ape_5.6-2
#> [65] tweenr_2.0.2            nlme_3.1-160             labeling_0.4.2
#> [69] rlang_1.0.6              lifecycle_1.0.3          pkgload_1.2.4
#> [73] downloader_0.4           filelock_1.0.2           sandwich_3.0-2
#> [77] rsvd_1.0.5              rprojroot_2.0.3          BiocFileCache_2.6.0
#> [81] graph_1.76.0             Matrix_1.5-3             GSVA_1.46.0
#> [85] zoo_1.8-11               beeswarm_0.4.0          Rhdf5lib_1.20.0
#> [89] artificer.ranges_1.3.4  bitops_1.0-7              viridisLite_0.4.1
#> [93] getPass_0.2-2            debugme_1.1.0             artificer.se_1.3.4
#> [97] Biostrings_2.66.0        blob_1.2.3               EnrichmentBrowser_2.20
#> [101] qvalue_2.30.0          coin_1.4-2               stringr_1.4.0
#> [105] beachmat_2.14.0        scales_1.2.1              gridGraphics_0.5-1
#> [109] magrittr_2.0.3          plyr_1.8.8               GSEABase_1.60.0
#> [113] compiler_4.2.0          dqrng_0.3.0               scatterpie_0.1.8
#> [117] cli_3.4.1              XVector_0.38.0            KEGGgraph_1.58.0
#> [121] MASS_7.3-58.1          tidyselect_1.2.0          ArtifactDB_1.9.5
#> [125] yaml_2.3.5              GOSemSim_2.24.0           stringi_1.7.6
#> [129] ggrepel_0.9.2          grid_4.2.0                highr_0.9
#> [133] tools_4.2.0             parallel_4.2.0            locfit_1.5-9.6
#> [137] metapod_1.6.0           gridExtra_2.3              fastmatch_1.1-3
#> [141] digest_0.6.29          BiocManager_1.30.19        bluster_1.8.0
#>
```

## Multiome tutorial - MultiAssayExperiment

```
#> [145] gp.auth_1.7.0          httr_1.4.3           AnnotationDbi_1.60.0
#> [149] brio_1.1.3            XML_3.99-0.12        splines_4.2.0
#> [153] statmod_1.4.37       tidytree_0.4.1        scater_1.26.1
#> [157] ArchR_1.0.2          ggplotify_0.1.0       xtable_1.8-4
#> [161] jsonlite_1.8.3        tidygraph_1.2.2       ggrepel_0.9.9
#> [165] R6_2.5.1             pillar_1.8.1         htmltools_0.5.3
#> [169] fastmap_1.1.0        clusterProfiler_4.6.0 BiocParallel_1.32.3
#> [173] codetools_0.2-18      fgsea_1.24.0          gp.version_1.5.0
#> [177] utf8_1.2.2           lattice_0.20-45      bslib_0.4.1
#> [181] curl_4.3.2           genomitory.schemas_0.99.0 ggbeeswarm_0.6.0
#> [185] magick_2.7.3          GO.db_3.16.0          survival_3.4-0
#> [189] dsassembly_1.7.6     dsdb.schemas_0.99.1   rmarkdown_2.18
#> [193] munsell_0.5.0         rhdf5_2.42.0          GenomeInfoDbData_1.2.9
#> [197] reshape2_1.4.4         gtable_0.3.1
```