Multiome tutorial - MultiAssayExperiment

Xiaosai Yao

September 4th, 2022

Package

epiregulon 1.0.14

Contents

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

library(epiregulon)
```

3 Data preparation

Single cell preprocessing needs to 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"]]
rownames(PeakMatrix) <- rowData(PeakMatrix)$idx

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

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

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@24c22e4f42
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      *
#>       ...      ...               ...    ...
#>   [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   0.5621008    50578 0.3047296
#> 132       2 chr1 818831 819331     21 AL645608.2   0.5425891    90102 0.3528375
#> 33        5 chr1 858735 859235     17  LINC01128   0.5785191    30938 0.2655123
#> 6         6 chr1 869650 870150     14 AL669831.2   0.6190875   108540 0.1774618
#> 76       10 chr1 920987 921487     19     FAM41C   0.6474293    50587 0.1262456
#> 433      10 chr1 920987 921487     28      PERM1   0.5072458    57540 0.4428137
```

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
#> 1       1     2     5-mC chr1 817121 817621     19 FAM41C 0.5621008    50578
#> 2       1    22 AML1-ETO chr1 817121 817621     19 FAM41C 0.5621008    50578
#> 3       1    25       AR chr1 817121 817621     19 FAM41C 0.5621008    50578
#> 4       1    49     ATF1 chr1 817121 817621     19 FAM41C 0.5621008    50578
#> 5       1    50     ATF2 chr1 817121 817621     19 FAM41C 0.5621008    50578
#> 6       1    51     ATF3 chr1 817121 817621     19 FAM41C 0.5621008    50578
#>         FDR
#> 1 0.3047296
#> 2 0.3047296
#> 3 0.3047296
#> 4 0.3047296
#> 5 0.3047296
#> 6 0.3047296
```

4.5 Network pruning (highly recommended)

```
pruned.regulon <- calculateJointProbability(expMatrix = GeneExpressionMatrix,
                                            exp_assay = "counts",
                                            peakMatrix = PeakMatrix,
                                            peak_assay = "counts",
                                            regulon = regulon[regulon$tf %in%
                                                        c("NKX2-1","GATA6","FOXA1","FOXA2", "AR","SOX2"),],
                                            clusters = GeneExpressionMatrix$Clusters,
                                            exp_cutoff = 0.5,
                                            peak_cutoff = 0)
#> AR
```

```
#> FOXA1
#> FOXA2
#> GATA6
#> NKX2-1
#> SOX2
```

4.6 Add Weights

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 = pruned.regulon,
                        sce = GeneExpressionMatrix,
                        cluster_factor = "Clusters",
                        exprs_values = "counts",
                        block_factor = NULL,
                        corr = TRUE,
                        MI = FALSE,
                        BPPARAM = BiocParallel::MulticoreParam())
```

```
head(regulon.w)
#>    tf   target triple_prop_all      weight
#> 1  AR     AAAS    0.0020497054  0.59468240
#> 3  AR    AAGAB    0.0115295926 -0.60605052
#> 6  AR     AAK1    0.0125544453 -0.14260301
#> 9  AR    AAMDC    0.0005124263  0.04598338
#> 11 AR     AARS    0.0184473482  0.73007603
#> 14 AR AASDHPPT    0.0030745580 -0.43858397
```

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 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 = GeneExpressionMatrix,
                                   regulon = regulon.w,
                                   mode = "weight",
```

```
                                        method = "weightedMean",
                                        assay = "counts")
```

4.8 Perform differential activity

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

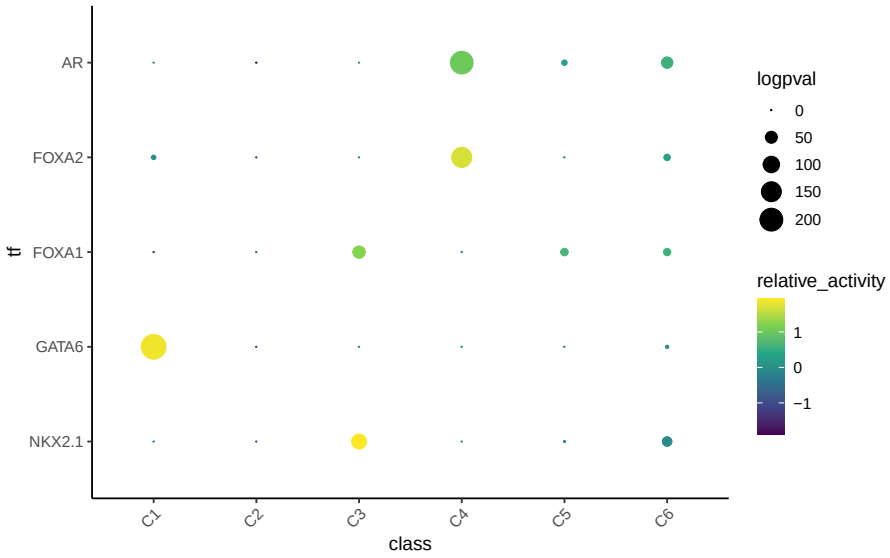
Take the top TFs

```
markers.sig <- getSigGenes(markers, topgenes = 5 )
#> Using a logFC cutoff of 0.6 for class C1
#> Using a logFC cutoff of -0.1 for class C2
#> Using a logFC cutoff of 0.5 for class C3
#> Using a logFC cutoff of 0.6 for class C4
#> Using a logFC cutoff of 0.1 for class C5
#> Using a logFC cutoff of 0.6 for class C6
```

4.9 Visualize the results

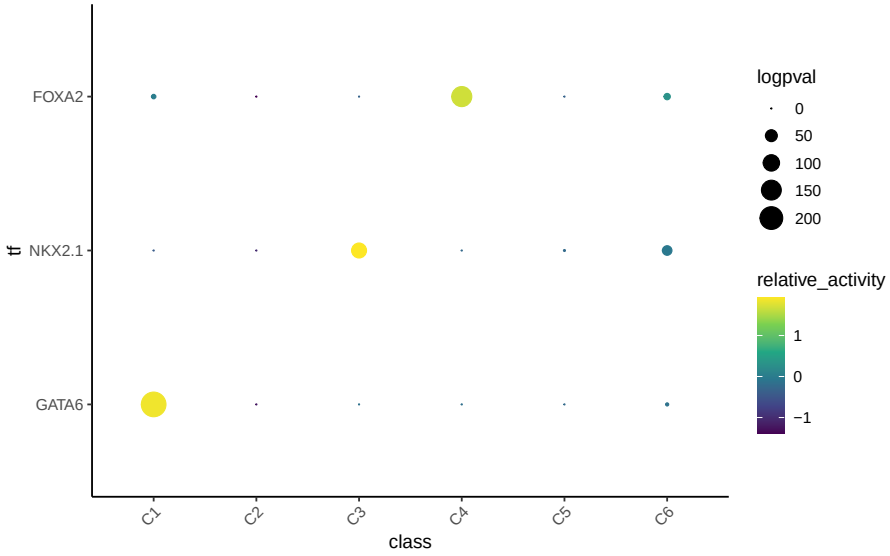
First visualize the known differential TFs by bubble plot

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



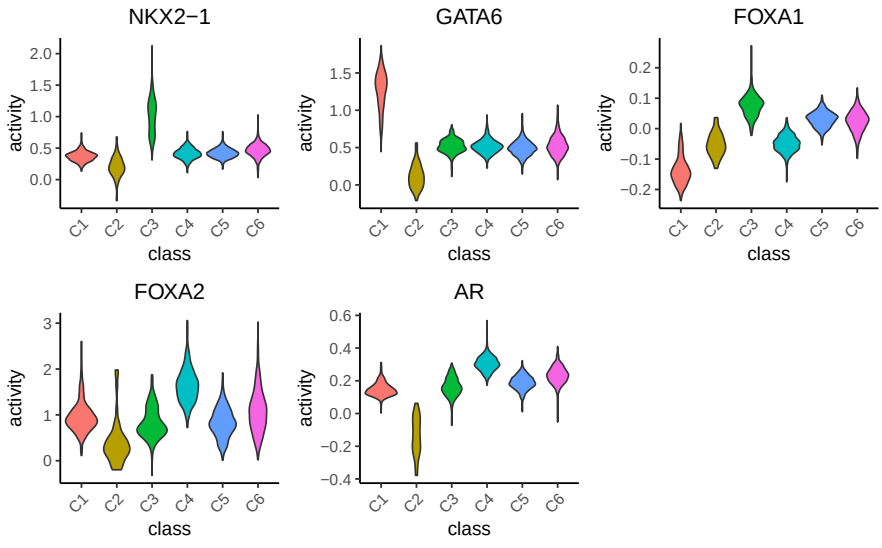
Then visualize the most differential TFs by clusters

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



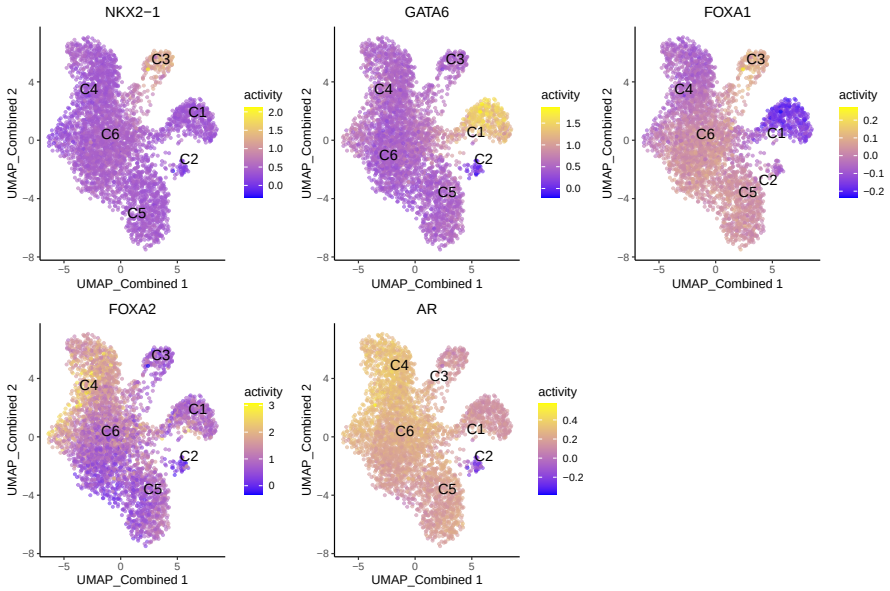
Visualize the known differential TFs by violin plot. Note there is no activity calculated for SOX2 because the expression of SOX2 is 0 in all cells.

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



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=1,
                ncol = 3)
#> SOX2 not found in activity matrix. Excluded from plots
```

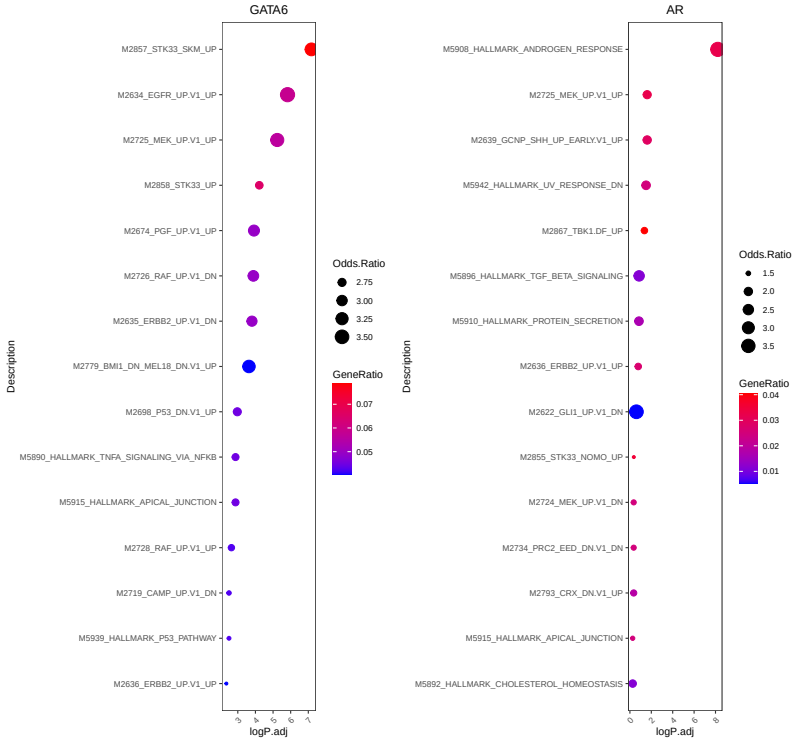


4.10 Geneset enrichment

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 2022-09-29 16:49:42
C6 <- EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb", cat = "C6",
                                     gene.id.type = "SYMBOL" )
#> Using cached version from 2022-09-29 16:49:46

#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",
```

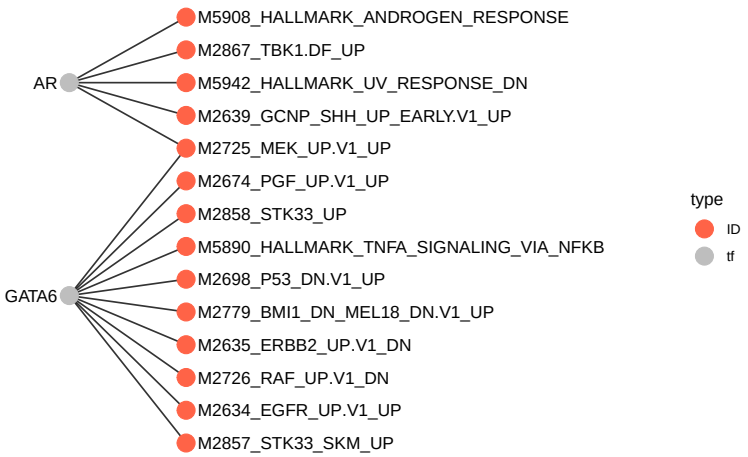
```
                                corr_cutoff = 0.5,
                                genesets = gs.list)

    #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 can calculate joint probabilities for every cluster

```
differential.regulon <- calculateJointProbability(expMatrix = GeneExpressionMatrix,
                                    exp_assay = "counts",
                                    peakMatrix = PeakMatrix,
                                    peak_assay = "counts",
                                    regulon = regulon[regulon$tf %in%
                                                c("NKX2-1","GATA6","FOXA1","FOXA2", "AR","SOX2"),],
                                    clusters = GeneExpressionMatrix$Clusters,
                                    exp_cutoff = 0.5,
                                    peak_cutoff = 0,
                                    aggregate = FALSE)
#> compute joint probability for all trios
#> AR
#> FOXA1
#> FOXA2
#> GATA6
#> NKX2-1
#> SOX2

head(differential.regulon)
#>         idxATAC idxTF tf  chr  start    end idxRNA     target      corr
#> AR.146        2    25 AR chr1 818831 819331     21 AL645608.2 0.5425891
#> AR.157        5    25 AR chr1 858735 859235     17  LINC01128 0.5785191
#> AR.553       10    25 AR chr1 920987 921487     28      PERM1 0.5072458
#> AR.1330      16    25 AR chr1 941458 941958     17  LINC01128 0.6455583
#> AR.1655      22    25 AR chr1 960317 960817     33       AGRN 0.7003204
#> AR.1657      22    25 AR chr1 960317 960817     28      PERM1 0.6699558
#>         distance        FDR  p_val_all z_score_all triple_prop_all  p_val_C1
```

```
#> AR.146     90102 0.35283752 0.05291648   1.9356019    0.0007686395 1.0000000
#> AR.157     30938 0.26551234 0.27935725  -1.0817644    0.0002562132 1.0000000
#> AR.553     57540 0.44281367 0.20227118   1.2751076    0.0002562132 1.0000000
#> AR.1330   113661 0.12932563 0.01897831   2.3459568    0.0043556239 1.0000000
#> AR.1655    57301 0.05761597 0.06348267   1.8557958    0.0043556239 0.7783302
#> AR.1657    18210 0.09322409 0.35029092   0.9340251    0.0002562132 1.0000000
#>         z_score_C1 triple_prop_C1 p_val_C2 z_score_C2 triple_prop_C2 p_val_C3
#> AR.146   0.0000000    0.000000000        1          0              0 1.000000
#> AR.157   0.0000000    0.000000000        1          0              0 1.000000
#> AR.553   0.0000000    0.000000000        1          0              0 1.000000
#> AR.1330  0.0000000    0.000000000        1          0              0 0.213221
#> AR.1655 -0.2814957    0.004524887        1          0              0 1.000000
#> AR.1657  0.0000000    0.000000000        1          0              0 1.000000
#>         z_score_C3 triple_prop_C3  p_val_C4 z_score_C4 triple_prop_C4  p_val_C5
#> AR.146    0.000000     0.00000000 1.0000000  0.0000000    0.000000000 1.0000000
#> AR.157    0.000000     0.00000000 0.4513104  0.7532322    0.001234568 1.0000000
#> AR.553    0.000000     0.00000000 0.1150565  1.5758669    0.001234568 1.0000000
#> AR.1330   1.244757     0.01357466 0.3039119  1.0280806    0.004938272 0.3559544
#> AR.1655   0.000000     0.00000000 0.1583781  1.4105474    0.003703704 0.1199547
#> AR.1657   0.000000     0.00000000 1.0000000  0.0000000    0.000000000 1.0000000
#>         z_score_C5 triple_prop_C5   p_val_C6 z_score_C6 triple_prop_C6
#> AR.146   0.0000000    0.000000000 0.02650766  2.2186924    0.001851852
#> AR.157   0.0000000    0.000000000 0.63411848 -0.4759381    0.000000000
#> AR.553   0.0000000    0.000000000 1.00000000  0.0000000    0.000000000
#> AR.1330  0.9231013    0.005235602 0.13975394  1.4767080    0.003703704
#> AR.1655  1.5549640    0.006544503 0.10016525  1.6440530    0.004320988
#> AR.1657  0.0000000    0.000000000 0.11668979  1.5688204    0.000617284
```

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
#>  [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] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] msigdbr_7.5.1               BiocStyle_2.25.0
#>  [3] rmarkdown_2.16              epiregulon_1.0.14
#>  [5] SingleCellExperiment_1.19.1 SummarizedExperiment_1.27.3
#>  [7] Biobase_2.57.1              GenomicRanges_1.49.1
#>  [9] GenomeInfoDb_1.33.7         IRanges_2.31.2
#> [11] S4Vectors_0.35.4            BiocGenerics_0.43.4
#> [13] MatrixGenerics_1.9.1        matrixStats_0.62.0
#>
#> loaded via a namespace (and not attached):
#>   [1] rappdirs_0.3.3              R.methodsS3_1.8.2
#>   [3] tidyr_1.2.1                 ggplot2_3.3.6
#>   [5] bit64_4.0.5                 knitr_1.40
#>   [7] irlba_2.3.5.1               DelayedArray_0.23.2
#>   [9] R.utils_2.12.0              data.table_1.14.2
#>  [11] KEGGREST_1.37.3             RCurl_1.98-1.9
#>  [13] generics_0.1.3              ScaledMatrix_1.5.1
#>  [15] callr_3.7.0                 cowplot_1.1.1
#>  [17] usethis_2.1.6               RSQLite_2.2.18
#>  [19] shadowtext_0.1.2            artificer.mae_1.3.4
#>  [21] bit_4.0.4                   enrichplot_1.17.2
#>  [23] base64url_1.4               gp.cache_1.7.1
#>  [25] httpuv_1.6.6                assertthat_0.2.1
#>  [27] genomitory_2.1.5            viridis_0.6.2
#>  [29] xfun_0.31                   babelgene_22.9
#>  [31] evaluate_0.16               promises_1.2.0.1
#>  [33] fansi_1.0.3                 dbplyr_2.2.1
#>  [35] Rgraphviz_2.41.1            igraph_1.3.5
#>  [37] DBI_1.1.3                   purrr_0.3.4
#>  [39] ellipsis_0.3.2              dplyr_1.0.10
#>  [41] backports_1.4.1             bookdown_0.29
#>  [43] annotate_1.75.0             sparseMatrixStats_1.9.0
#>  [45] artificer.matrix_1.3.7      vctrs_0.4.1
#>  [47] remotes_2.4.2               entropy_1.3.1
#>  [49] dsdb.plus_1.3.2             cachem_1.0.6
#>  [51] withr_2.5.0                 ggforce_0.4.1
#>  [53] checkmate_2.1.0             metacommons_1.9.0
#>  [55] treeio_1.21.2               MultiAssayExperiment_1.23.9
#>  [57] prettyunits_1.1.1           scran_1.25.1
#>  [59] cluster_2.1.3               DOSE_3.23.2
#>  [61] BiocBaseUtils_0.99.12       ape_5.6-2
#>  [63] lazyeval_0.2.2              crayon_1.5.1
#>  [65] labeling_0.4.2              edgeR_3.39.6
#>  [67] pkgconfig_2.0.3             tweenr_2.0.2
#>  [69] nlme_3.1-159                vipor_0.4.5
#>  [71] pkgload_1.2.4               devtools_2.4.3
#>  [73] rlang_1.0.6                 lifecycle_1.0.1
#>  [75] artificer.schemas_0.99.2    downloader_0.4
```

```
#>  [77] filelock_1.0.2             artificer.base_1.3.18
#>  [79] BiocFileCache_2.5.0        rsvd_1.0.5
#>  [81] rprojroot_2.0.3            polyclip_1.10-0
#>  [83] GSVA_1.45.3                graph_1.75.0
#>  [85] Matrix_1.4-1               aplot_0.1.7
#>  [87] Rhdf5lib_1.19.2            beeswarm_0.4.0
#>  [89] processx_3.5.3             png_0.1-7
#>  [91] viridisLite_0.4.1          artificer.ranges_1.3.4
#>  [93] bitops_1.0-7               artificer.se_1.3.4
#>  [95] getPass_0.2-2              R.oo_1.25.0
#>  [97] gson_0.0.9                 rhdf5filters_1.9.0
#>  [99] EnrichmentBrowser_2.27.0   Biostrings_2.65.6
#> [101] blob_1.2.3                 DelayedMatrixStats_1.19.0
#> [103] stringr_1.4.0              qvalue_2.29.0
#> [105] gridGraphics_0.5-1         beachmat_2.13.4
#> [107] scales_1.2.1               memoise_2.0.1
#> [109] GSEABase_1.59.0            magrittr_2.0.3
#> [111] plyr_1.8.7                 zlibbioc_1.43.0
#> [113] compiler_4.2.0             scatterpie_0.1.8
#> [115] dqrng_0.3.0                RColorBrewer_1.1-3
#> [117] KEGGgraph_1.57.0           cli_3.3.0
#> [119] XVector_0.37.1             patchwork_1.1.2
#> [121] ps_1.7.0                   ArtifactDB_1.9.5
#> [123] MASS_7.3-58.1              tidyselect_1.1.2
#> [125] stringi_1.7.6              yaml_2.3.5
#> [127] GOSemSim_2.23.0            BiocSingular_1.13.1
#> [129] locfit_1.5-9.6             ggrepel_0.9.1
#> [131] grid_4.2.0                 fastmatch_1.1-3
#> [133] tools_4.2.0                parallel_4.2.0
#> [135] rstudioapi_0.13            bluster_1.7.0
#> [137] AUCell_1.19.1              metapod_1.5.0
#> [139] gridExtra_2.3              farver_2.1.1
#> [141] ggraph_2.0.6               digest_0.6.29
#> [143] BiocManager_1.30.18        shiny_1.7.2
#> [145] Rcpp_1.0.9                 scuttle_1.7.4
#> [147] later_1.3.0                gp.auth_1.7.0
#> [149] httr_1.4.3                 AnnotationDbi_1.59.1
#> [151] colorspace_2.0-3           brio_1.1.3
#> [153] XML_3.99-0.11              fs_1.5.2
#> [155] splines_4.2.0              yulab.utils_0.0.5
#> [157] statmod_1.4.37             tidytree_0.4.1
#> [159] scater_1.25.7              graphlayouts_0.8.2
#> [161] ArchR_1.0.2                ggplotify_0.1.0
#> [163] sessioninfo_1.2.2          xtable_1.8-4
#> [165] jsonlite_1.8.2             ggtree_3.5.3
#> [167] tidygraph_1.2.2            ggfun_0.0.7
#> [169] testthat_3.1.4             ShadowArray_1.7.1
#> [171] R6_2.5.1                   pillar_1.7.0
#> [173] htmltools_0.5.3            mime_0.12
#> [175] glue_1.6.2                 fastmap_1.1.0
#> [177] clusterProfiler_4.5.2      BiocParallel_1.31.12
```

```
#> [179] BiocNeighbors_1.15.1        codetools_0.2-18
#> [181] fgsea_1.23.2                gp.version_1.5.0
#> [183] pkgbuild_1.3.1              utf8_1.2.2
#> [185] lattice_0.20-45             tibble_3.1.7
#> [187] curl_4.3.2                  genomitory.schemas_0.99.0
#> [189] ggbeeswarm_0.6.0            artificer.sce_1.3.4
#> [191] GO.db_3.16.0                limma_3.53.10
#> [193] dsassembly_1.7.6            dsdb.schemas_0.99.1
#> [195] desc_1.4.1                  munsell_0.5.0
#> [197] DO.db_2.9                   rhdf5_2.41.1
#> [199] GenomeInfoDbData_1.2.9      HDF5Array_1.25.2
#> [201] reshape2_1.4.4              gtable_0.3.1
```