Multiome tutorial - MultiAssayExperiment

Xiaosai Yao

18 December 2022

Package

epiregulon 1.0.22

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_github(repo ='xiaosaiyao/epiregulon')

library(epiregulon)
```

3 Data preparation

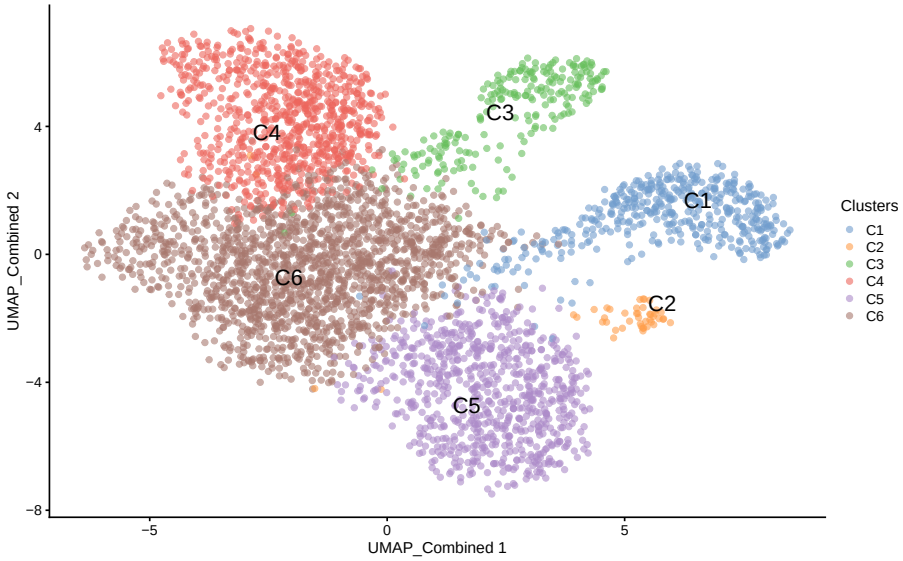
Single cell preprocessing needs to performed by user's favorite methods prior to using Epiregulon. The following components are required: 1. Peak matrix from scATAC-seq 2. Gene expression matrix from either paired or unpaired scRNA-seq. RNA-seq integration needs to be performed for unpaired dataset. 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"]]

# 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
#> 4         4 chr1 827287 827787     14 AL669831.2   0.5118844    66177
#> 336       7 chr1 904515 905015     26     KLHL17   0.5234782    53567
#> 338       9 chr1 920452 920952     26     KLHL17   0.5289008    37630
#> 529      12 chr1 924540 925040     30       HES4   0.6339207    72939
#> 113      20 chr1 958517 959017     20 AL645608.6   0.5290174    53484
#> 88       22 chr1 960317 960817     19     FAM41C   0.7241886    89917
#>             FDR
#> 4   0.42414016
#> 336 0.39525030
#> 338 0.38124047
#> 529 0.14493813
#> 113 0.38100171
#> 88  0.03549889
```

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
#> 161       4     2 5-mC
#> 162       4     8 AFF1
#> 163       4     9 AFF4
#> 164       4    10 AGO1
#> 165       4    11 AGO2
#> 166       4    14  AHR
```

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       4     2 5-mC chr1 827287 827787     14 AL669831.2 0.5118844    66177
#> 2       4     8 AFF1 chr1 827287 827787     14 AL669831.2 0.5118844    66177
#> 3       4     9 AFF4 chr1 827287 827787     14 AL669831.2 0.5118844    66177
#> 4       4    10 AGO1 chr1 827287 827787     14 AL669831.2 0.5118844    66177
#> 5       4    11 AGO2 chr1 827287 827787     14 AL669831.2 0.5118844    66177
#> 6       4    14  AHR chr1 827287 827787     14 AL669831.2 0.5118844    66177
#>         FDR
#> 1 0.4241402
#> 2 0.4241402
#> 3 0.4241402
#> 4 0.4241402
#> 5 0.4241402
#> 6 0.4241402
```

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(TF, RE) * P(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(TF,

RE) * P(TG) and the probability otherwise is 1- P(TF, RE) * P(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_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.

```
pruned.regulon <- pruneRegulon(expMatrix = GeneExpressionMatrix,
                               exp_assay = "counts",
                               peakMatrix = PeakMatrix,
                               peak_assay = "counts",
                               test = "chi.sq",
                               regulon,
                               clusters = GeneExpressionMatrix$Clusters,
                               prune_value = "pval",
                               regulon_cutoff = 0.05
                               )
```

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, 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.

```
regulon.w <- addWeights(regulon = pruned.regulon,
                        expMatrix  = GeneExpressionMatrix,
                        exp_assay  = "counts",
                        peakMatrix = PeakMatrix,
                        peak_assay = "counts",
                        clusters = GeneExpressionMatrix$Clusters,
                        block_factor = NULL,
                        tf_re.merge = TRUE,
                        method = "corr")
```

```
head(regulon.w)
#>       idxATAC idxTF   tf  chr   start     end idxRNA     target      corr
#> 1515       22     5 ADNP chr1  960317  960817     28      PERM1 0.5690398
```

```
#> 15684     121     5 ADNP chr1 1375168 1375668     61      CCNL2 0.5382674
#> 16617     122     5 ADNP chr1 1375717 1376217     56       CPTP 0.5248150
#> 34405     655     5 ADNP chr1 8061169 8061669    210 AL034417.4 0.5144690
#> 34406     655     5 ADNP chr1 8061169 8061669    208    TNFRSF9 0.8141961
#> 37253     787     5 ADNP chr1 9290024 9290524    229      SPSB1 0.9643174
#>       distance          FDR     pval_all    pval_C1   pval_C2   pval_C3
#> 1515     18210 2.820260e-01 1.062648e-01 0.83574296       NaN 0.8157231
#> 15684    13273 3.579263e-01 5.648278e-03 0.08295649 0.8827557 0.1570487
#> 16617    50716 3.919597e-01 7.073687e-04 0.73658475       NaN 0.8490927
#> 34405    69836 4.176845e-01 1.376811e-01 0.77532454       NaN       NaN
#> 34406   120104 3.345260e-03 5.790391e-04 0.10494073       NaN       NaN
#> 37253      368 2.861965e-08 1.634348e-09 0.46442485       NaN       NaN
#>            pval_C4   pval_C5      pval_C6 stats_all   stats_C1   stats_C2
#> 1515  7.803215e-01       NaN 6.149301e-05  2.608910 0.04299061        NaN
#> 15684 7.455996e-01 0.3650078 8.284839e-02  7.659224 3.00600553 0.02174941
#> 16617 1.544434e-05 0.8146175 4.442433e-02 11.469785 0.11315113        NaN
#> 34405 8.751367e-01 0.7299816 1.881518e-04  2.203686 0.08146298        NaN
#> 34406 6.168540e-19 0.8212453 8.848119e-01 11.842183 2.62880407        NaN
#> 37253 8.424454e-01       NaN 9.011359e-01 36.367149 0.53520957        NaN
#>         stats_C3    stats_C4   stats_C5    stats_C6    padj_all padj_C1 padj_C2
#> 1515  0.05431199  0.07778525        NaN 16.05610618 1.000000000       1     NaN
#> 15684 2.00241619  0.10526591 0.82058960  3.00811874 1.000000000       1       1
#> 16617 0.03620503 18.68187805 0.05497778  4.04036050 1.000000000       1     NaN
#> 34405        NaN  0.02469211 0.11912852 13.94585792 1.000000000       1     NaN
#> 34406        NaN 79.01354997 0.05105053  0.02098793 1.000000000       1     NaN
#> 37253        NaN  0.03950810        NaN  0.01543225 0.005200939       1     NaN
#>       padj_C3      padj_C4 padj_C5 padj_C6    weight
#> 1515        1 1.000000e+00     NaN       1 0.4561281
#> 15684       1 1.000000e+00       1       1 0.1261201
#> 16617       1 1.000000e+00       1       1 0.1015220
#> 34405     NaN 1.000000e+00       1       1 0.4693631
#> 34406     NaN 1.974347e-12       1       1 0.8466490
#> 37253     NaN 1.000000e+00     NaN       1 0.9980748
```

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,...,n} $weights_i$ is the weight of TF and target i

```
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")
```

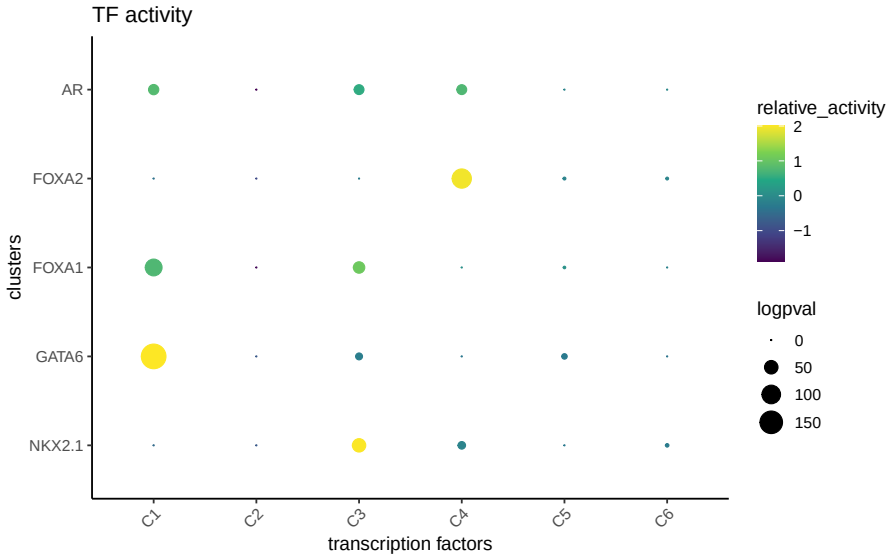
Take the top TFs

```
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

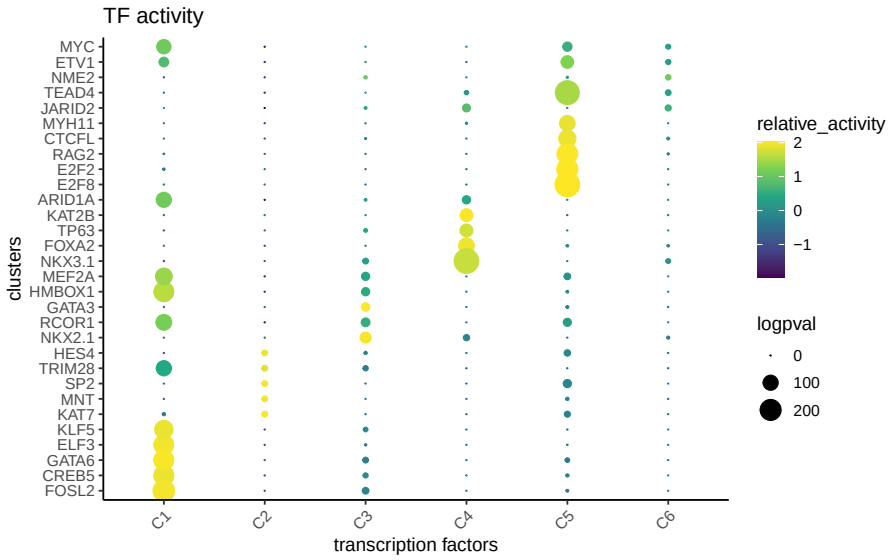
First visualize the known differential TFs by bubble plot

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



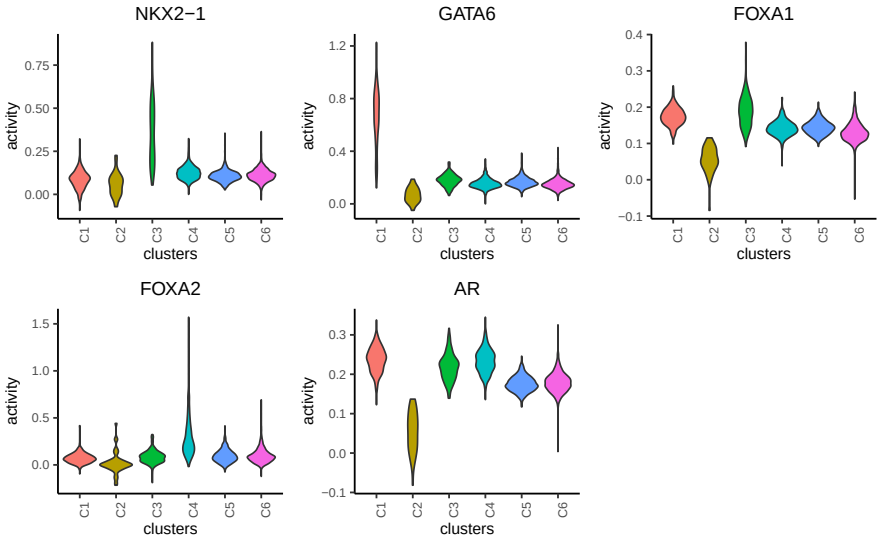
Then visualize the most differential TFs by clusters

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



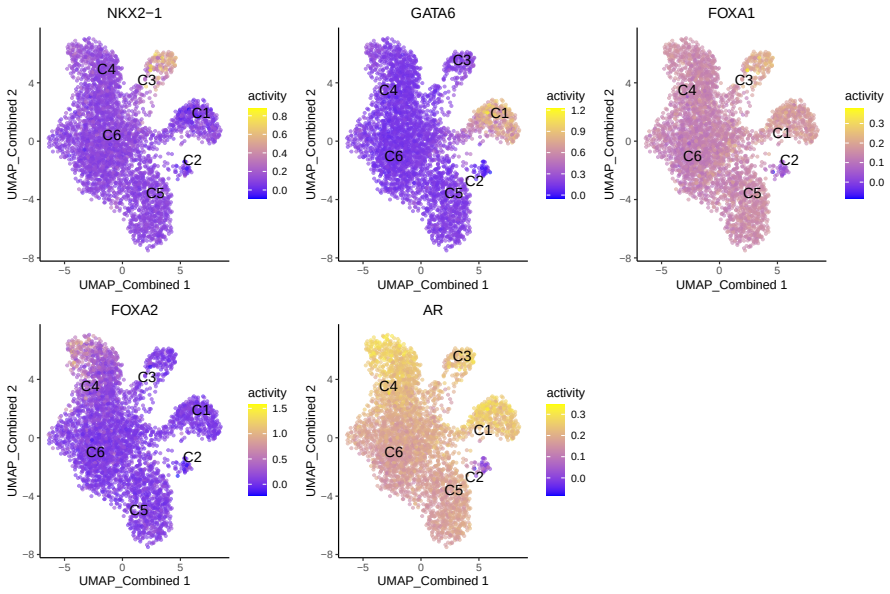
Visualize the known differential TFs by violin plot. Note there is no activity calculated for SOX2 because the expression of SOX2 is 0 in all cells.

```
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
```



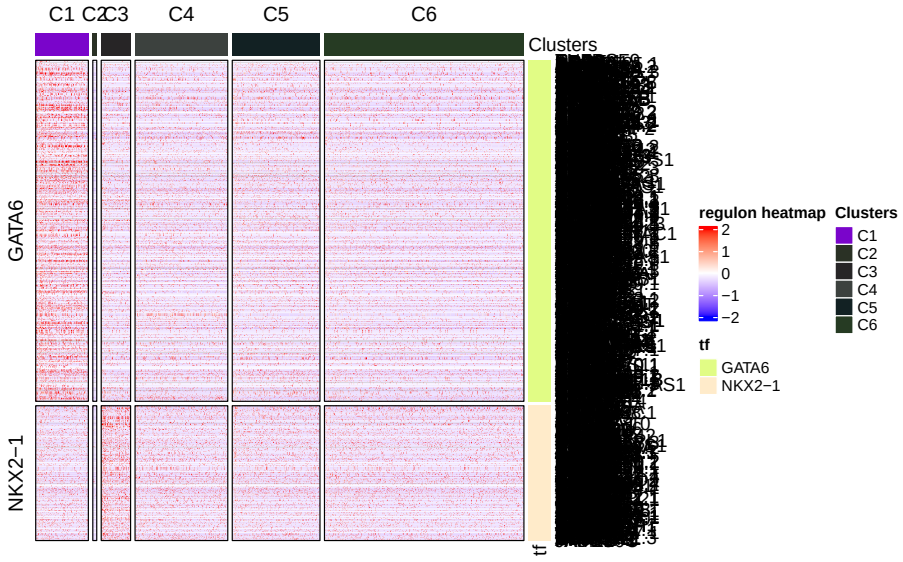
Visualize the known differential TFs by UMAP

```
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
```

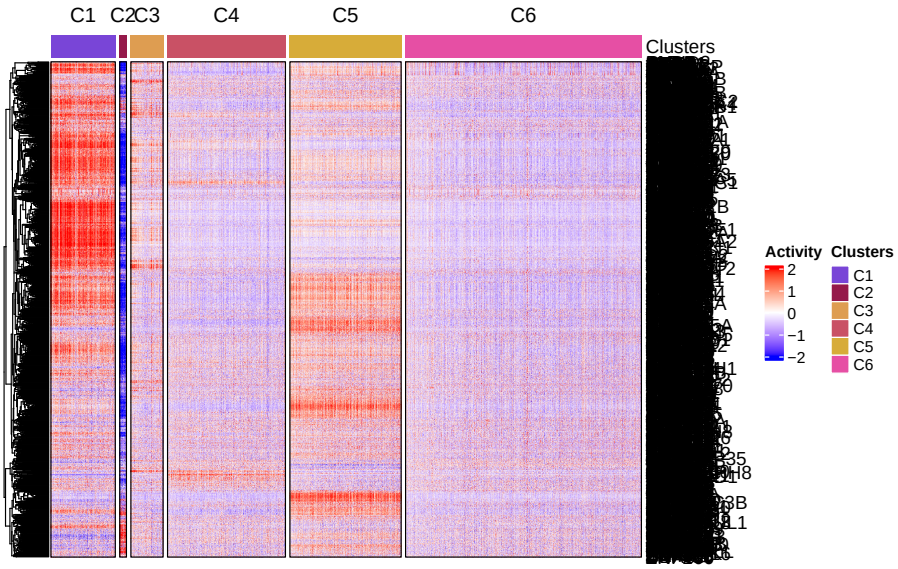


Visualize the gene expression of the regulons by heatmap

```
plotHeatmapRegulon(sce=GeneExpressionMatrix,
                   tfs=c("GATA6","NKX2-1"),
                   regulon=regulon.w,
                   regulon_cutoff=0.1,
                   downsample=1000,
                   cell_attributes="Clusters",
                   col_gap="Clusters",
                   exprs_values="counts",
                   name="regulon heatmap")
```



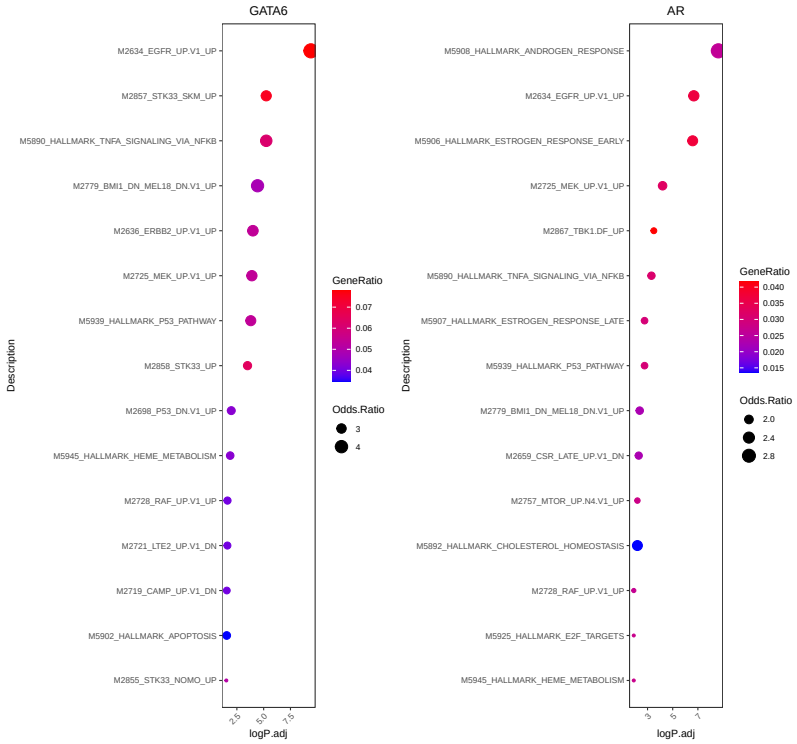

```
plotHeatmapActivity(activity=score.combine,
                    sce=GeneExpressionMatrix,
                    tfs=rownames(score.combine),
                    downsample=5000,
                    cell_attributes="Clusters",
                    col_gap="Clusters",
                    name = "Activity")
```



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.

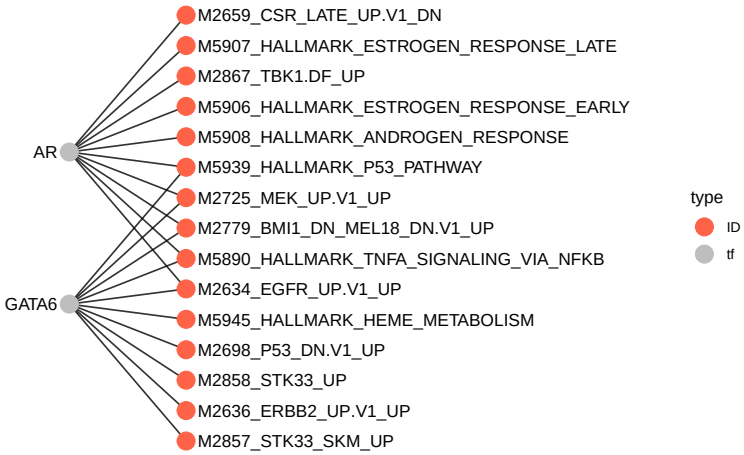
```
#retrieve genesets
H <- EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb", cat = "H",
                                    gene.id.type = "SYMBOL" )
#> Using cached version from 2022-12-18 20:57:58
C6 <- EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb", cat = "C6",
                                     gene.id.type = "SYMBOL" )
#> Using cached version from 2022-12-18 20:58:03

#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)
#> GATA6
#>
#> AR

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



4.11 Network analysis

We can visualize the genesets as a network

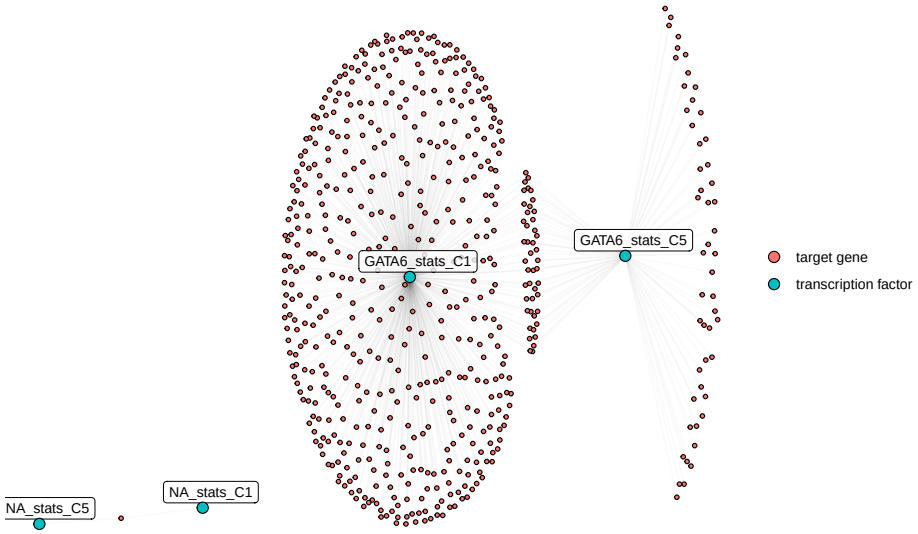
```
plotGseaNetwork(tf = names(enrichresults),
                enrichresults = enrichresults,
                p.adj_cutoff = 0.1,
                ntop_pathways = 10)
```



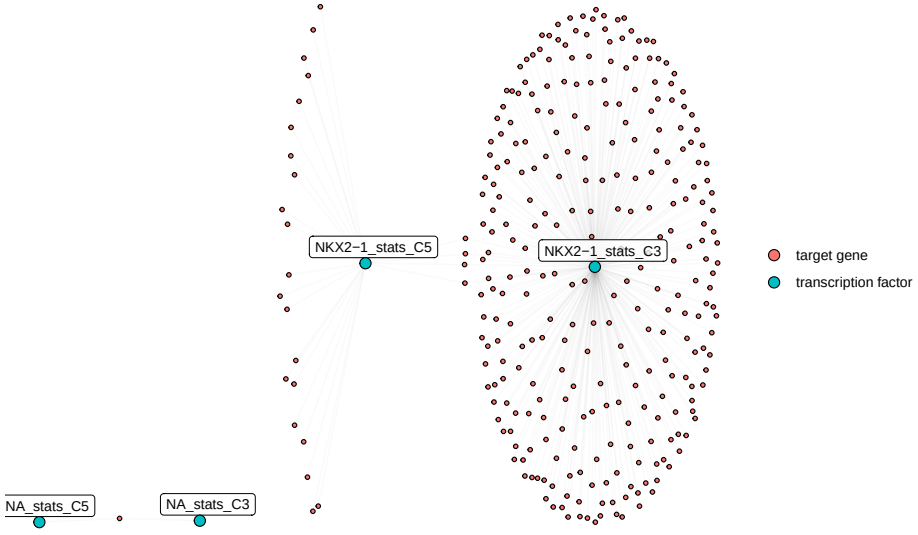
4.12 Differential networks

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")
```




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

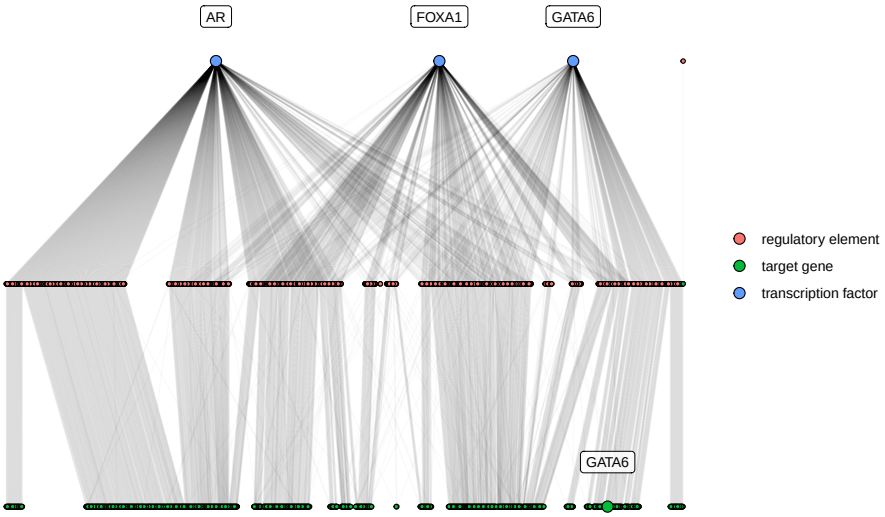


We can also visualize how transcription factors relate to other transcription factors in each cluster.

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

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

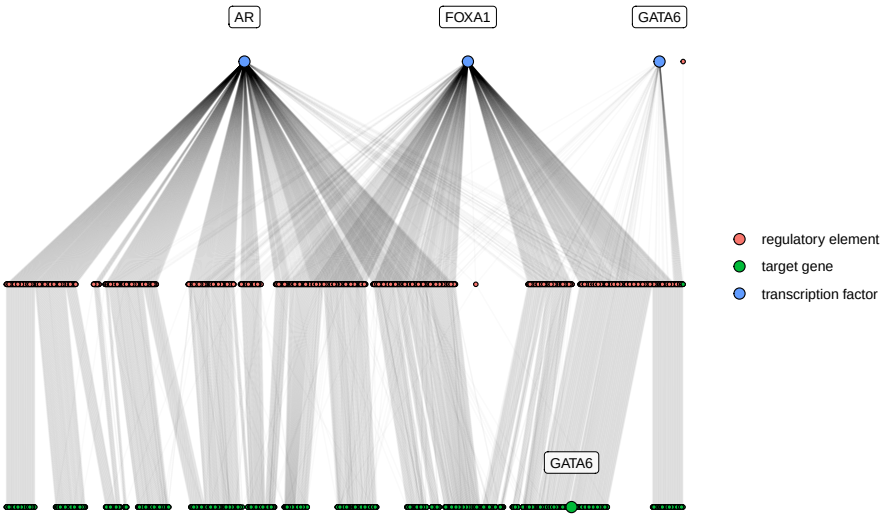
```
                          tfs_to_highlight = c("GATA6","FOXA1","AR")) + ggplot2::ggtitle ("C1")
```

C1



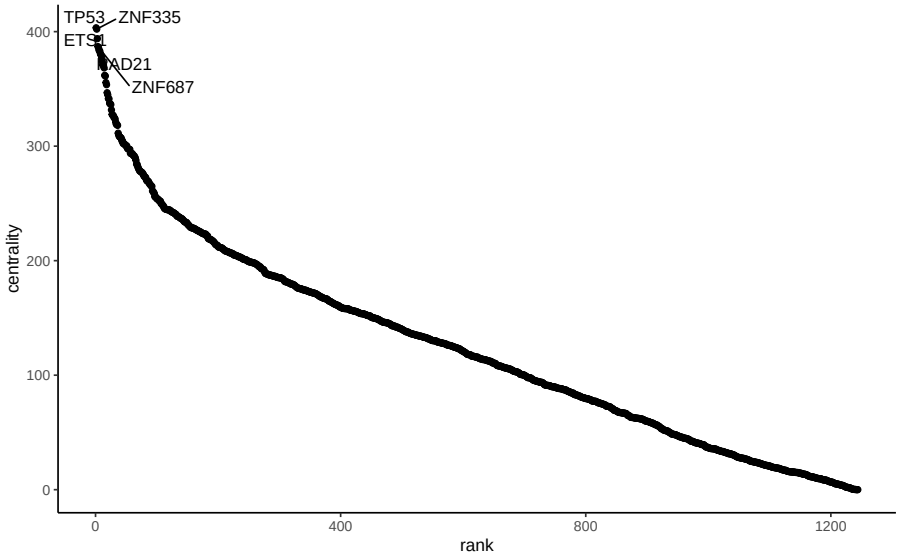

```
plotEpiregulonNetwork(C5_network,
                      layout = "sugiyama",
                      tfs_to_highlight = c("GATA6","FOXA1","AR")) + ggplot2::ggtitle ("C5")
```

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.

```
# 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
#>  [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] ggplot2_3.4.0               msigdbr_7.5.1
#>  [3] epiregulon_1.0.22           SingleCellExperiment_1.20.0
#>  [5] SummarizedExperiment_1.29.1 Biobase_2.58.0
#>  [7] GenomicRanges_1.50.2        GenomeInfoDb_1.34.4
#>  [9] IRanges_2.32.0              S4Vectors_0.36.1
#> [11] BiocGenerics_0.44.0         MatrixGenerics_1.10.0
#> [13] matrixStats_0.63.0          BiocStyle_2.26.0
#>
#> loaded via a namespace (and not attached):
#>   [1] utf8_1.2.2                  tidyselect_1.2.0
#>   [3] RSQLite_2.2.19              AnnotationDbi_1.60.0
#>   [5] grid_4.2.0                  BiocParallel_1.32.4
#>   [7] scatterpie_0.1.8            munsell_0.5.0
#>   [9] ScaledMatrix_1.6.0          base64url_1.4
#>  [11] codetools_0.2-18            statmod_1.4.37
#>  [13] scran_1.26.1                withr_2.5.0
#>  [15] GOSemSim_2.24.0             colorspace_2.0-3
#>  [17] genomitory_2.1.6            filelock_1.0.2
#>  [19] knitr_1.40                  DOSE_3.23.3
#>  [21] labeling_0.4.2              KEGGgraph_1.58.2
#>  [23] GenomeInfoDbData_1.2.9      polyclip_1.10-4
#>  [25] bit64_4.0.5                 farver_2.1.1
#>  [27] rhdf5_2.42.0                downloader_0.4
#>  [29] treeio_1.22.0               vctrs_0.5.1
#>  [31] generics_0.1.3              gson_0.0.9
#>  [33] xfun_0.31                   BiocFileCache_2.6.0
#>  [35] R6_2.5.1                    doParallel_1.0.17
#>  [37] graphlayouts_0.8.4          ggbeeswarm_0.7.1
#>  [39] clue_0.3-63                 rsvd_1.0.5
#>  [41] gp.version_1.5.0            locfit_1.5-9.6
#>  [43] artificer.matrix_1.3.7      gridGraphics_0.5-1
#>  [45] fgsea_1.24.0                bitops_1.0-7
#>  [47] rhdf5filters_1.10.0         cachem_1.0.6
#>  [49] DelayedArray_0.24.0         assertthat_0.2.1
#>  [51] scales_1.2.1                ggraph_2.1.0
#>  [53] enrichplot_1.18.3           beeswarm_0.4.0
#>  [55] gtable_0.3.1                beachmat_2.14.0
#>  [57] Cairo_1.6-0                 metacommons_1.9.0
#>  [59] tidygraph_1.2.2             rlang_1.0.6
#>  [61] splines_4.2.0               GlobalOptions_0.1.2
#>  [63] lazyeval_0.2.2              dsdb.schemas_0.99.1
```

```
#>  [65] checkmate_2.1.0            gp.auth_1.7.0
#>  [67] BiocManager_1.30.19        yaml_2.3.5
#>  [69] reshape2_1.4.4             backports_1.4.1
#>  [71] qvalue_2.30.0              clusterProfiler_4.6.0
#>  [73] tools_4.2.0                bookdown_0.30
#>  [75] ggplotify_0.1.0            RColorBrewer_1.1-3
#>  [77] artificer.base_1.3.19      MultiAssayExperiment_1.24.0
#>  [79] Rcpp_1.0.9                 plyr_1.8.8
#>  [81] sparseMatrixStats_1.10.0   zlibbioc_1.44.0
#>  [83] purrr_0.3.5                RCurl_1.98-1.9
#>  [85] artificer.ranges_1.3.4     GetoptLong_1.0.5
#>  [87] viridis_0.6.2              cowplot_1.1.1
#>  [89] ShadowArray_1.7.1          ggrepel_0.9.2
#>  [91] cluster_2.1.3              magrittr_2.0.3
#>  [93] data.table_1.14.6          magick_2.7.3
#>  [95] circlize_0.4.15            patchwork_1.1.2
#>  [97] evaluate_0.19              GSVA_1.46.0
#>  [99] xtable_1.8-4               HDO.db_0.99.0
#> [101] XML_3.99-0.13              artificer.schemas_0.99.2
#> [103] artificer.sce_1.3.4        gridExtra_2.3
#> [105] shape_1.4.6                compiler_4.2.0
#> [107] scater_1.26.1              tibble_3.1.8
#> [109] shadowtext_0.1.2           crayon_1.5.1
#> [111] gp.cache_1.7.1             htmltools_0.5.4
#> [113] ggfun_0.0.9                aplot_0.1.9
#> [115] tidyr_1.2.1                ArtifactDB_1.9.5
#> [117] DBI_1.1.3                  tweenr_2.0.2
#> [119] dbplyr_2.2.1               ComplexHeatmap_2.14.0
#> [121] MASS_7.3-58.1              rappdirs_0.3.3
#> [123] babelgene_22.9             Matrix_1.5-3
#> [125] cli_3.4.1                  artificer.mae_1.3.4
#> [127] genomitory.schemas_0.99.0  parallel_4.2.0
#> [129] metapod_1.6.0              igraph_1.3.5
#> [131] pkgconfig_2.0.3            getPass_0.2-2
#> [133] dsdb.plus_1.3.2            scuttle_1.8.3
#> [135] foreach_1.5.2              ggtree_3.6.2
#> [137] annotate_1.76.0            vipor_0.4.5
#> [139] dqrng_0.3.0                ArchR_1.0.2
#> [141] XVector_0.38.0             yulab.utils_0.0.5
#> [143] stringr_1.4.0              digest_0.6.29
#> [145] graph_1.76.0               Biostrings_2.66.0
#> [147] fastmatch_1.1-3            rmarkdown_2.18
#> [149] tidytree_0.4.1             artificer.se_1.3.5
#> [151] edgeR_3.40.1               DelayedMatrixStats_1.20.0
#> [153] GSEABase_1.60.0            curl_4.3.2
#> [155] rjson_0.2.21               nlme_3.1-161
#> [157] lifecycle_1.0.3            jsonlite_1.8.4
#> [159] Rhdf5lib_1.20.0            dsassembly_1.7.6
#> [161] BiocNeighbors_1.16.0       viridisLite_0.4.1
#> [163] limma_3.54.0               fansi_1.0.3
#> [165] pillar_1.8.1               lattice_0.20-45
```

```
#> [167] GO.db_3.16.0                KEGGREST_1.38.0
#> [169] fastmap_1.1.0               httr_1.4.3
#> [171] glue_1.6.2                  png_0.1-8
#> [173] iterators_1.0.14            bluster_1.8.0
#> [175] bit_4.0.5                   Rgraphviz_2.42.0
#> [177] ggforce_0.4.1               stringi_1.7.6
#> [179] HDF5Array_1.26.0            BiocBaseUtils_1.1.0
#> [181] EnrichmentBrowser_2.28.0    blob_1.2.3
#> [183] BiocSingular_1.14.0         memoise_2.0.1
#> [185] dplyr_1.0.10                ape_5.6-2
#> [187] irlba_2.3.5.1
```