Hematopoeisis tutorial - ArchR project

Shang-Yang and Xiaosai Yao

12 November 2022

Package

epiregulon 1.0.16

Contents

1 Introduction

Gene regulatory networks model the underlying gene regulation hierarchies that drive gene expression and observed phenotypes. The main function of the epiregulon R package is to infer TF activity in single cells by constructing a gene regulatory network (regulons). This is achieved through integration of scATAC-seq and scRNA-seq data and incorporation of public bulk TF ChIP-seq data. Links between regulatory elements and their target genes are established by computing correlations between chromatin accessibility and gene expressions.

Current prerequisite for running epiregulon is a ArchR project with pre-computed peak and gene matrices. It is also expected that LSI dimensionality reduction and integration with an scRNA-seq dataset has been performed. The scATAC-seq experiment can be either paired or unpaired with the scRNA-seq dataset as long as they were already integrated by ArchR. The final output of epiregulon is a matrix of TF activities where rows are individual TFs and columns are single cell indexes.

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 to derive a more stringent set of P2G correlations compared to ArchR.

In this vignette we demonstrate the workflow of epiregulon along with some visualization functionalities using the tutorial datasets from ArchR development team. In this dataset, scRNAseq and scATACseq were unpaired and integrated by the `addGeneIntegrationMatrix` function.

2 Installation

Epiregulon is currently available on R/dev

```
library(epiregulon)
```

If you would like to 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

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 scATAC-seq 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 scRNA-seq and scATAC-seq using addCombinedDims

To verify that all the necessary matrices are present,

```
library(ArchR,verbose = FALSE)
archR_project_path <- "/gstore/project/lineage/sam/heme_GRN/OUTPUT"
proj <- loadArchRProject(path = archR_project_path, showLogo = FALSE)
#> Successfully loaded ArchRProject!
getAvailableMatrices(proj)
#> [1] "GeneIntegrationMatrix" "GeneScoreMatrix"       "PeakMatrix"
#> [4] "TileMatrix"
```

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 genome mm10 are available

```
grl <- getTFMotifInfo(genome = "hg19")
#> redirecting from 'GMTY156:hg19_motif_bed_granges@REVISION-5' to 'GMTY156:hg19_motif_bed_granges@83b9ff4cd
head(grl)
#> GRangesList object of length 6:
#> $`5-hmC`
#> GRanges object with 22860 ranges and 0 metadata columns:
#>           seqnames        ranges strand
#>              <Rle>     <IRanges>  <Rle>
#>       [1]     chr1   10001-10685      *
#>       [2]     chr1   13362-13694      *
#>       [3]     chr1   29631-29989      *
#>       [4]     chr1   40454-40754      *
#>       [5]     chr1 135395-135871      *
#>       ...      ...           ...    ...
#>   [22856]     chrM   15303-15326      *
#>   [22857]     chrM   15328-16172      *
#>   [22858]     chrM   16174-16183      *
#>   [22859]     chrM   16186-16224      *
#>   [22860]     chrM   16226-16492      *
#>   -------
#>   seqinfo: 25 sequences from an unspecified genome; no seqlengths
#>
#> ...
#> <5 more elements>
```

4.2 Link ATACseq peaks to target genes

Next, we compute peak to gene correlations using the calculateP2G function from ArchR package. The user would need to supply a path to an ArchR project that already contains the peak matrix, gene expression matrix and Latent semantic indexing (LSI) dimensionality reduction. The example project shown here utilizes the tutorial datasets provided by the ArchR development team.

```
# path to ArchR project
p2g <- calculateP2G(ArchR_path = archR_project_path)
#> Setting ArchRLogging = FALSE
#> Using ArchR to compute peak to gene links...
#> 2022-11-12 13:13:57 : Getting Available Matrices, 0 mins elapsed.
#> 2022-11-12 13:13:59 : Filtered Low Prediction Score Cells (684 of 10250, 0.067), 0.012 mins elapsed.
#> 2022-11-12 13:14:00 : Computing KNN, 0.023 mins elapsed.
#> 2022-11-12 13:14:00 : Identifying Non-Overlapping KNN pairs, 0.028 mins elapsed.
#> 2022-11-12 13:14:06 : Identified 492 Groupings!, 0.132 mins elapsed.
#> 2022-11-12 13:14:06 : Getting Group RNA Matrix, 0.132 mins elapsed.
#> 2022-11-12 13:15:02 : Getting Group ATAC Matrix, 1.057 mins elapsed.
#> 2022-11-12 13:16:03 : Normalizing Group Matrices, 2.08 mins elapsed.
#> 2022-11-12 13:16:10 : Finding Peak Gene Pairings, 2.2 mins elapsed.
#> 2022-11-12 13:16:11 : Computing Correlations, 2.208 mins elapsed.
#> 2022-11-12 13:16:17 : Completed Peak2Gene Correlations!, 2.318 mins elapsed.
head(p2g)
#>   idxATAC  chr  start    end idxRNA    target Correlation distance          FDR
#> 1       7 chr1 801002 801502      2 LINC00115   0.8324892    38350 6.106975e-125
#> 2      24 chr1 894453 894953      6    KLHL17   0.5261049     1264  5.377552e-35
#> 3      24 chr1 894453 894953     14  TNFRSF18   0.5125436   247386  5.592283e-33
#> 5      25 chr1 894960 895460      9     ISG15   0.6024232    53637  2.843810e-48
#> 4      25 chr1 894960 895460     14  TNFRSF18   0.5509001   246879  6.249324e-39
#> 6      36 chr1 934450 934950     17   B3GALT6   0.5289083   232929  2.005290e-35
```

Alternatively, users can now also supply peak, gene, and dimensional reduction matrices derived from a MultiAssayExperiment object. This is compatible with future GPSA multiome workflow. Epiregulon implements a custom algorithm to derive a more stringent set of P2G correlations compared to ArchR.

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

p2g <- calculateP2G(peakmatrix, expmatrix, reducedDim)

head(p2g)
```

4.3 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(p2g, grl, archR_project_path = archR_project_path)
#> Successfully loaded ArchRProject!
#> Computing overlap...
#> Success!
head(overlap)
#>      idxATAC idxTF      tf
#> 1018       7    35    ARNT
#> 1019       7    50    ATF2
#> 1020       7    55    ATF7
#> 1021       7    76    BCL6
#> 1022       7    80    BCOR
#> 1023       7    82 BHLHE40
```

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, overlap, aggregate=TRUE)
head(regulon)
#>                         tf  target      corr
#> 1 8-Hydroxydeoxyguanosine A2M-AS1 0.6186957
#> 2                    AFF1 A2M-AS1 0.6061886
#> 3                     AHR A2M-AS1 0.6061886
#> 4                AML1-ETO A2M-AS1 0.5986916
#> 5                      AR A2M-AS1 0.6061886
#> 6                  ARID1A A2M-AS1 0.6061886
```

Epiregulon outputs two different correlations. The first, termed "corr", is the correlation between chromatin accessibility of regulatory elements vs expression of target genes calculated by ArchR. The second, termed "weight", can be generated by the addWeights function, which compute the correlation between gene expressions of TF vs expressions of target genes, shown below. The user is required to supply the clustering or batch labels of the scRNA-seq dataset when running addWeights. "Weight" is the preferred metric for calculating activity.

load scRNA-seq data

```
sce <- readRDS("/gstore/project/lineage/sam/heme_GRN/scRNA-Granja-2019.rds")
```

Trim regulon for demonstration purposes

```
TFs <- c("FOXA1","GATA3","SOX9", "SPI1")
regulon <- regulon[which(regulon$tf %in% TFs),]
nrow(regulon)
#> [1] 19481

regulon.w <- addWeights(regulon = regulon,
                        expMatrix = sce,
                        clusters = sce$BioClassification,
                        block_factor = NULL,
                        method = "corr",
                        BPPARAM = BiocParallel::MulticoreParam())
#> adding weights using corr
#> calculating average expression across clusters...
#> computing correlation of the regulon...
#>
  |
  |                                                                      |   0%
  |
  |====================                                                  |  25%
  |
  |========================================                              |  50%
  |
  |============================================================          |  75%
  |
  |================================================================================| 100%
head(regulon.w)
#>         tf  target      corr      weight
#> 87   FOXA1 A2M-AS1 0.6411842 -0.09270487
#> 447  FOXA1  A4GALT 0.7511119 -0.04072622
#> 839  FOXA1   AAGAB 0.5867544  0.01432174
#> 1046 FOXA1    AAK1 0.6172998 -0.11678985
#> 1794 FOXA1   AAMDC 0.6182848  0.06322510
#> 2072 FOXA1    AAMP 0.5874550 -0.14155255
```

4.5 Calculate TF activity

Finally, the activities for a specific TF in each cell are computed by averaging the weighted expressions of target genes linked to the TF weighted.

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

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

```
score.combine <- calculateActivity(expMatrix = sce,
                                   regulon = regulon.w,
                                   mode = "weight",
                                   method = "weightedMean",
                                   exp_assay = "logcounts")
```

```
#> calculating TF activity from regulon using weightedmean
head(score.combine[,1:10])
#>       CD34_32_R5:AAACCTGAGTATCGAA-1 CD34_32_R5:AAACCTGAGTCGTTTG-1
#> FOXA1                   0.007260328                   0.014892169
#> GATA3                  -0.021553769                  -0.027724326
#> SOX9                   -0.001871930                  -0.007102871
#> SPI1                   -0.008189045                   0.004348497
#>       CD34_32_R5:AAACCTGGTTCCACAA-1 CD34_32_R5:AAACGGGAGCTTCGCG-1
#> FOXA1                  0.0022521031                   0.005113500
#> GATA3                 -0.0160899155                  -0.019234281
#> SOX9                  -0.0007590259                  -0.002876271
#> SPI1                  -0.0074112998                  -0.008703520
#>       CD34_32_R5:AAACGGGAGGGAGTAA-1 CD34_32_R5:AAACGGGAGTTACGGG-1
#> FOXA1                  0.0027397744                  0.0019127790
#> GATA3                 -0.0133900050                 -0.0126762382
#> SOX9                  -0.0005292432                  0.0006287226
#> SPI1                  -0.0056637788                 -0.0091275573
#>       CD34_32_R5:AAACGGGCAAGTAATG-1 CD34_32_R5:AAACGGGCAAGTTCTG-1
#> FOXA1                 -0.0009537837                   0.001628196
#> GATA3                 -0.0069966788                  -0.012493376
#> SOX9                   0.0026079655                   0.000342098
#> SPI1                  -0.0064179739                  -0.006363953
#>       CD34_32_R5:AAACGGGCACAGACAG-1 CD34_32_R5:AAACGGGGTAACGTTC-1
#> FOXA1                   0.003800214                   0.003976918
#> GATA3                  -0.023720200                  -0.018734750
#> SOX9                   -0.003789301                  -0.004355170
#> SPI1                   -0.013049582                  -0.005607091
```

4.6 Differential TF activity test

We can next determine which TFs exhibit differential activities across cell clusters/groups via the findDifferentialActivity function. This function depends on findMarkers function from scran package and allow the same parameters.

```
da_list <- findDifferentialActivity(activity_matrix = score.combine,
                                    groups = sce$BioClassification,
                                    pval.type = "some",
                                    direction = "up",
                                    test.type = "t")
```

getSigGenes compiles the different test results into a single dataframe and enables user to supply their desired cutoffs for significance and variable to order by.

```
markers <- getSigGenes(da_list, fdr_cutoff = 0.05)
#> Using a logFC cutoff of 0 for class 01_HSC
#> Using a logFC cutoff of 0 for class 02_Early.Eryth
#> Using a logFC cutoff of 0 for class 03_Late.Eryth
#> Using a logFC cutoff of 0 for class 04_Early.Baso
#> Using a logFC cutoff of 0 for class 05_CMP.LMPP
#> Using a logFC cutoff of 0 for class 06_CLP.1
```

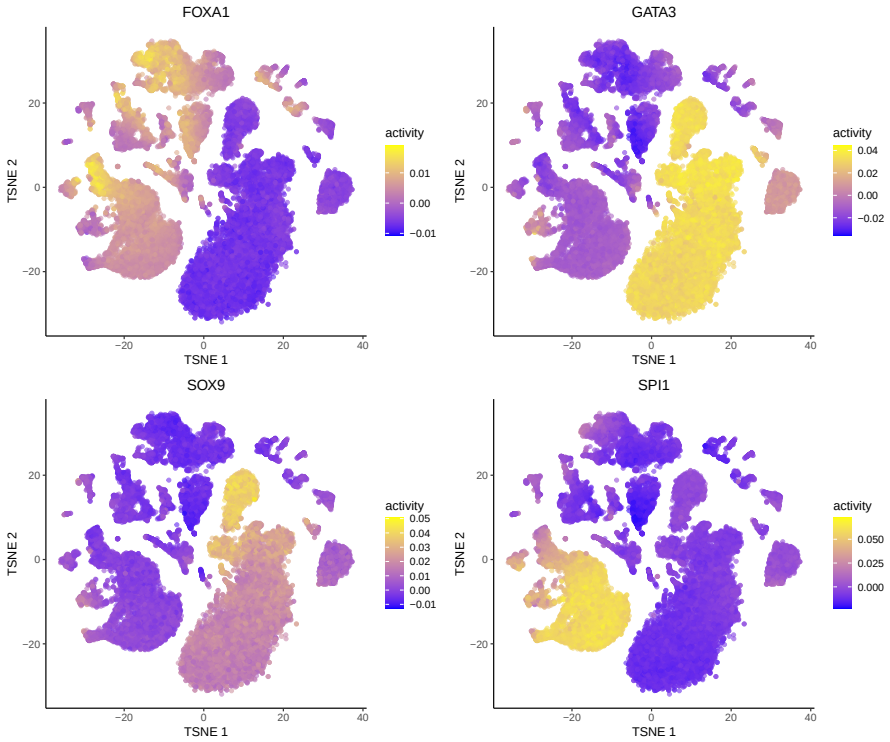
```
#> Using a logFC cutoff of 0 for class 07_GMP
#> Using a logFC cutoff of 0 for class 08_GMP.Neut
#> Using a logFC cutoff of 0 for class 09_pDC
#> Using a logFC cutoff of 0 for class 10_cDC
#> Using a logFC cutoff of 0.1 for class 11_CD14.Mono.1
#> Using a logFC cutoff of 0.1 for class 12_CD14.Mono.2
#> Using a logFC cutoff of 0.1 for class 13_CD16.Mono
#> Using a logFC cutoff of 0 for class 14_Unk
#> Using a logFC cutoff of 0 for class 15_CLP.2
#> Using a logFC cutoff of 0 for class 16_Pre.B
#> Using a logFC cutoff of 0 for class 17_B
#> Using a logFC cutoff of 0 for class 18_Plasma
#> Using a logFC cutoff of 0 for class 19_CD8.N
#> Using a logFC cutoff of 0 for class 20_CD4.N1
#> Using a logFC cutoff of 0 for class 21_CD4.N2
#> Using a logFC cutoff of 0 for class 22_CD4.M
#> Using a logFC cutoff of 0 for class 23_CD8.EM
#> Using a logFC cutoff of 0 for class 24_CD8.CM
#> Using a logFC cutoff of 0 for class 25_NK
#> Using a logFC cutoff of 0 for class 26_Unk
head(markers)
#>         p.value           FDR summary.logFC          class    tf
#> 1  3.883871e-03  1.553549e-02  0.0003006128         01_HSC FOXA1
#> 2 4.128807e-146 1.651523e-145  0.0032349512 02_Early.Eryth FOXA1
#> 3 8.450047e-102 3.380019e-101  0.0047956807  03_Late.Eryth FOXA1
#> 4  9.893183e-43  3.957273e-42  0.0044446429  04_Early.Baso FOXA1
#> 5 6.144314e-242 2.457726e-241  0.0049082073    05_CMP.LMPP FOXA1
#> 6 9.919074e-102 3.967629e-101  0.0051236055       06_CLP.1 FOXA1
```

4.7 Visualizing TF activities

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

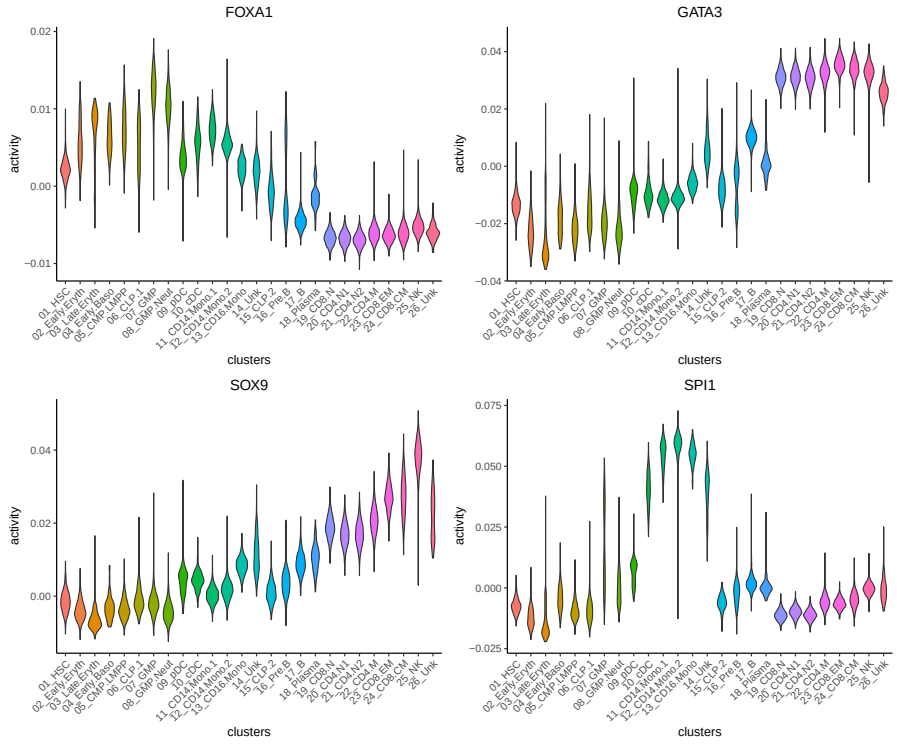
tSNE or UMAP plots:

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



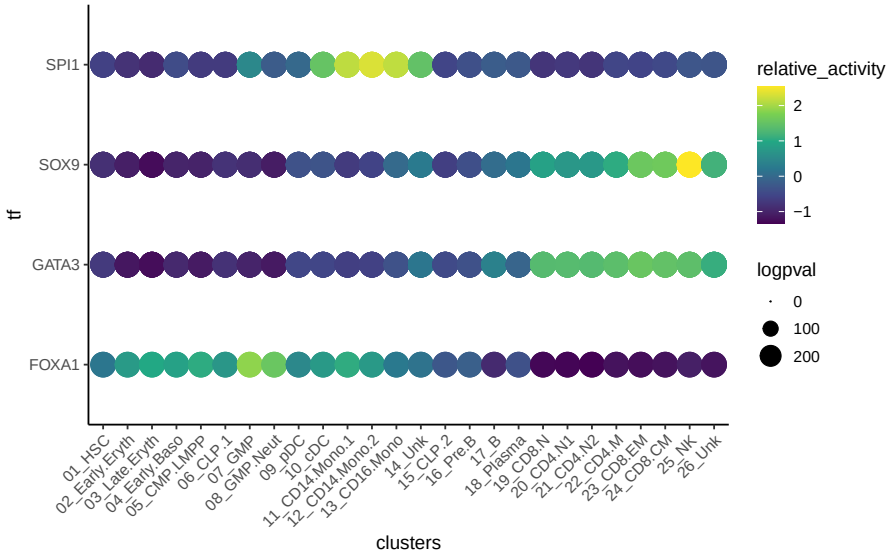
Violin plots:

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



Bubble plot:

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



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
#>
#> Random number generation:
#>  RNG:     L'Ecuyer-CMRG
#>  Normal:  Inversion
#>  Sample:  Rejection
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C         LC_TIME=C            LC_COLLATE=C
#>  [5] LC_MONETARY=C        LC_MESSAGES=C        LC_PAPER=C           LC_NAME=C
#>  [9] LC_ADDRESS=C         LC_TELEPHONE=C       LC_MEASUREMENT=C     LC_IDENTIFICATION=C
#>
#> attached base packages:
#>  [1] parallel  grid      stats4    stats     graphics  grDevices utils     datasets
#>  [9] methods   base
#>
#> other attached packages:
#>  [1] nabor_0.5.0                 rhdf5_2.42.0                Rcpp_1.0.9
#>  [4] Matrix_1.5-3                data.table_1.14.4           stringr_1.4.0
#>  [7] plyr_1.8.8                  magrittr_2.0.3              gtable_0.3.1
#> [10] gtools_3.9.3                gridExtra_2.3               ArchR_1.0.2
#> [13] ggplot2_3.4.0               msigdbr_7.5.1               epiregulon_1.0.16
#> [16] SingleCellExperiment_1.20.0 SummarizedExperiment_1.29.1 Biobase_2.58.0
#> [19] GenomicRanges_1.50.1        GenomeInfoDb_1.34.3         IRanges_2.32.0
#> [22] S4Vectors_0.36.0            BiocGenerics_0.44.0         MatrixGenerics_1.10.0
#> [25] matrixStats_0.62.0          BiocStyle_2.26.0
#>
#> loaded via a namespace (and not attached):
#>   [1] utf8_1.2.2                 tidyselect_1.2.0           RSQLite_2.2.18
#>   [4] AnnotationDbi_1.60.0       BiocParallel_1.32.1        scatterpie_0.1.8
#>   [7] munsell_0.5.0              ScaledMatrix_1.6.0         base64url_1.4
#>  [10] codetools_0.2-18           statmod_1.4.37             scran_1.26.0
#>  [13] withr_2.5.0                colorspace_2.0-3           GOSemSim_2.24.0
#>  [16] genomitory_2.1.6           filelock_1.0.2             knitr_1.40
#>  [19] rstudioapi_0.13            DOSE_3.23.3                labeling_0.4.2
#>  [22] KEGGgraph_1.58.0           GenomeInfoDbData_1.2.9     polyclip_1.10-4
#>  [25] bit64_4.0.5                farver_2.1.1               downloader_0.4
#>  [28] treeio_1.22.0              vctrs_0.5.0                generics_0.1.3
#>  [31] gson_0.0.9                 xfun_0.31                  BiocFileCache_2.6.0
#>  [34] R6_2.5.1                   graphlayouts_0.8.3         ggbeeswarm_0.6.0
#>  [37] rsvd_1.0.5                 gp.version_1.5.0           locfit_1.5-9.6
```

```
#>  [40] artificer.matrix_1.3.7      gridGraphics_0.5-1          bitops_1.0-7
#>  [43] rhdf5filters_1.10.0         cachem_1.0.6                fgsea_1.24.0
#>  [46] DelayedArray_0.24.0         assertthat_0.2.1            scales_1.2.1
#>  [49] ggraph_2.1.0                enrichplot_1.18.0           beeswarm_0.4.0
#>  [52] beachmat_2.14.0             Cairo_1.6-0                 metacommons_1.9.0
#>  [55] tidygraph_1.2.2             rlang_1.0.6                 splines_4.2.0
#>  [58] lazyeval_0.2.2              dsdb.schemas_0.99.1         checkmate_2.1.0
#>  [61] gp.auth_1.7.0               BiocManager_1.30.19         yaml_2.3.5
#>  [64] reshape2_1.4.4              backports_1.4.1             rsconnect_0.8.28
#>  [67] qvalue_2.30.0               clusterProfiler_4.6.0       tools_4.2.0
#>  [70] bookdown_0.30               ggplotify_0.1.0             RColorBrewer_1.1-3
#>  [73] artificer.base_1.3.19       MultiAssayExperiment_1.24.0 sparseMatrixStats_1.10.0
#>  [76] zlibbioc_1.44.0             purrr_0.3.5                 RCurl_1.98-1.9
#>  [79] artificer.ranges_1.3.4      viridis_0.6.2               cowplot_1.1.1
#>  [82] ShadowArray_1.7.1           ggrepel_0.9.2               cluster_2.1.3
#>  [85] tinytex_0.42                patchwork_1.1.2             evaluate_0.18
#>  [88] GSVA_1.46.0                 xtable_1.8-4                HDO.db_0.99.0
#>  [91] XML_3.99-0.12               artificer.schemas_0.99.2    artificer.sce_1.3.4
#>  [94] compiler_4.2.0              scater_1.26.0               tibble_3.1.8
#>  [97] shadowtext_0.1.2            crayon_1.5.1                gp.cache_1.7.1
#> [100] htmltools_0.5.3             ggfun_0.0.8                 aplot_0.1.8
#> [103] tidyr_1.2.1                 ArtifactDB_1.9.5            DBI_1.1.3
#> [106] tweenr_2.0.2                dbplyr_2.2.1                MASS_7.3-58.1
#> [109] rappdirs_0.3.3              babelgene_22.9              cli_3.4.1
#> [112] artificer.mae_1.3.4         genomitory.schemas_0.99.0   metapod_1.6.0
#> [115] igraph_1.3.5                pkgconfig_2.0.3             getPass_0.2-2
#> [118] dsdb.plus_1.3.2             scuttle_1.8.0               ggtree_3.6.2
#> [121] annotate_1.76.0             vipor_0.4.5                 dqrng_0.3.0
#> [124] XVector_0.38.0              yulab.utils_0.0.5           digest_0.6.29
#> [127] graph_1.76.0                Biostrings_2.66.0           rmarkdown_2.18
#> [130] fastmatch_1.1-3             tidytree_0.4.1              artificer.se_1.3.4
#> [133] edgeR_3.40.0                DelayedMatrixStats_1.20.0   GSEABase_1.60.0
#> [136] curl_4.3.2                  nlme_3.1-160                lifecycle_1.0.3
#> [139] jsonlite_1.8.3              Rhdf5lib_1.20.0             dsassembly_1.7.6
#> [142] BiocNeighbors_1.16.0        viridisLite_0.4.1           limma_3.54.0
#> [145] fansi_1.0.3                 pillar_1.8.1                lattice_0.20-45
#> [148] KEGGREST_1.38.0             fastmap_1.1.0               httr_1.4.3
#> [151] GO.db_3.16.0                glue_1.6.2                  png_0.1-7
#> [154] bluster_1.8.0               bit_4.0.4                   Rgraphviz_2.42.0
#> [157] ggforce_0.4.1               stringi_1.7.6               HDF5Array_1.26.0
#> [160] BiocBaseUtils_1.1.0         EnrichmentBrowser_2.28.0    blob_1.2.3
#> [163] BiocSingular_1.14.0         memoise_2.0.1               dplyr_1.0.10
#> [166] ape_5.6-2                   irlba_2.3.5.1
```