

prostate cancer archr tutorial

Xiaosai Yao

23 May 2023

Package

epiregulon 1.0.23

Contents

1	Introduction	2
2	Installation	2
3	Data preparation	2
3.1	Load ArchR project	2
3.2	Retrieve matrices from ArchR project	3
4	Quick start	5
4.1	Retrieve bulk TF ChIP-seq binding sites	5
4.2	Link ATAC-seq peaks to target genes	6
4.3	Add TF motif binding to peaks	7
4.4	Generate regulons	7
4.5	Calculate TF activity	10
4.6	Perform differential activity	12
4.7	Visualize the results	13
4.8	Geneset enrichment	16
4.9	Network analysis	18
5	Session Info	19

1 Introduction

This tutorial walks through an example of TF activity inference in unpaired scATAC-seq/scRNAseq of parental LNCaP cells treated with DMSO, Enzalutamide and Enza resistant cells. The dataset was taken from [Taavitsainen et al GSE168667](#) and [GSE168668](#).

2 Installation

Epiregulon is currently available on R/dev

Alternatively, you could install from gitlab

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

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 scATACseq by using addGroupCoverages > addReproduciblePeakSet > addPeakMatrix. See chapter 10 from ArchR manual 2. RNA-seq integration. a. For unpaired scATAC-seq, use addGeneIntegrationMatrix. See chapter 8 from ArchR manual b. For multiome data, use addGeneExpressionMatrix. See [multiome](#) tutorial 3. Perform dimensionality reduction from with either single modalities or joint scRNAseq and scATACseq using [addCombinedDims](#)

3.1 Load ArchR project

Copy this ArchR project into your own directory

```
archR_project_path <- "/gstore/project/lineage/prostate/GSE168667/OUTPUT/multiome/"
proj <- loadArchRProject(path = archR_project_path, showLogo = FALSE)
```

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

```
getAvailableMatrices(proj)
#> [1] "GeneIntegrationMatrix" "GeneScoreMatrix"           "MotifMatrix"
#> [4] "PeakMatrix"            "TileMatrix"
```

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

```
head(getReducedDims(proj, reducedDims = "iLSI_Combined")[,1:5])
#>          LSI1      LSI2      LSI3      LSI4
#> SRR13927735#TTATGTCTCCAGGTAT-1 -2.713935 -0.3677949 -0.4484238 -0.30645138
```

prostate cancer archr tutorial

```
#> SRR13927735#TATTGCTCATCAGAAA-1 -2.642781 -0.2767556 -0.9142714 -0.19675812
#> SRR13927735#TTCGATTGTAGGGTTG-1 -2.322865 -0.1543080 -1.4106049 -0.08891276
#> SRR13927735#CATTCAATTGGATGTT-1 -2.572976 -0.1917188 -1.0464294 -0.12660121
#> SRR13927735#ACGTTAGGTCAACTGT-1 -2.478552 -0.1776639 -1.1037295 -0.22976613
#> SRR13927735#AAATGCCAGCAATGG-1 -2.595352 -0.3803464 -0.7770309 -0.52431765
#>                                     LSI5
#> SRR13927735#TTATGTCTCCAGGTAT-1 -0.046845365
#> SRR13927735#TATTGCTCATCAGAAA-1 0.075746940
#> SRR13927735#TTCGATTGTAGGGTTG-1 0.019873276
#> SRR13927735#CATTCAATTGGATGTT-1 0.009947438
#> SRR13927735#ACGTTAGGTCAACTGT-1 -0.150097539
#> SRR13927735#AAATGCCAGCAATGG-1 -0.243074591
head(getEmbedding(proj, embedding = "UMAP_Combined"))
#>                                     iLSI_Combined#UMAP_Dimension_1
#> SRR13927735#TTATGTCTCCAGGTAT-1 -9.622903
#> SRR13927735#TATTGCTCATCAGAAA-1 -9.360211
#> SRR13927735#TTCGATTGTAGGGTTG-1 -8.617347
#> SRR13927735#CATTCAATTGGATGTT-1 -9.285448
#> SRR13927735#ACGTTAGGTCAACTGT-1 -8.809260
#> SRR13927735#AAATGCCAGCAATGG-1 -9.261216
#>                                     iLSI_Combined#UMAP_Dimension_2
#> SRR13927735#TTATGTCTCCAGGTAT-1 -0.2908237
#> SRR13927735#TATTGCTCATCAGAAA-1 -0.2892935
#> SRR13927735#TTCGATTGTAGGGTTG-1 -0.2154103
#> SRR13927735#CATTCAATTGGATGTT-1 -0.3267481
#> SRR13927735#ACGTTAGGTCAACTGT-1 -0.2168703
#> SRR13927735#AAATGCCAGCAATGG-1 0.3200356
```

3.2 Retrieve matrices from ArchR project

Retrieve gene expression and peak matrix from the ArchR project

```
GeneExpressionMatrix <- getMatrixFromProject(
  ArchRProj = proj,
  useMatrix = "GeneIntegrationMatrix",
  useSeqnames = NULL,
  verbose = TRUE,
  binarize = FALSE,
  threads = 1,
  logFile = "x"
)
#> ArchR logging to : x
#> If there is an issue, please report to github with logFile!
#> 2023-05-23 12:31:19.164121 : Organizing colData, 2.028 mins elapsed.
#> 2023-05-23 12:31:19.315365 : Organizing rowData, 2.031 mins elapsed.
#> 2023-05-23 12:31:19.318199 : Organizing rowRanges, 2.031 mins elapsed.
#> 2023-05-23 12:31:19.323502 : Organizing Assays (1 of 1), 2.031 mins elapsed.
#> 2023-05-23 12:31:27.760147 : Constructing SummarizedExperiment, 2.172 mins elapsed.
#> 2023-05-23 12:31:28.905656 : Finished Matrix Creation, 2.191 mins elapsed.
```

prostate cancer archr tutorial

```
PeakMatrix <- getMatrixFromProject(  
    ArchRProj = proj,  
    useMatrix = "PeakMatrix",  
    useSeqnames = NULL,  
    verbose = TRUE,  
    binarize = FALSE,  
    threads = 1,  
    logFile = "x"  
)  
#> 2023-05-23 12:32:31.956894 : Organizing colData, 1.051 mins elapsed.  
#> 2023-05-23 12:32:32.165551 : Organizing rowData, 1.054 mins elapsed.  
#> 2023-05-23 12:32:32.176176 : Organizing rowRanges, 1.054 mins elapsed.  
#> 2023-05-23 12:32:32.190128 : Organizing Assays (1 of 1), 1.055 mins elapsed.  
#> 2023-05-23 12:32:38.229308 : Constructing SummarizedExperiment, 1.155 mins elapsed.  
#> 2023-05-23 12:32:56.298519 : Finished Matrix Creation, 1.456 mins elapsed.
```

Convert gene expression matrix to SingleCellExperiment object

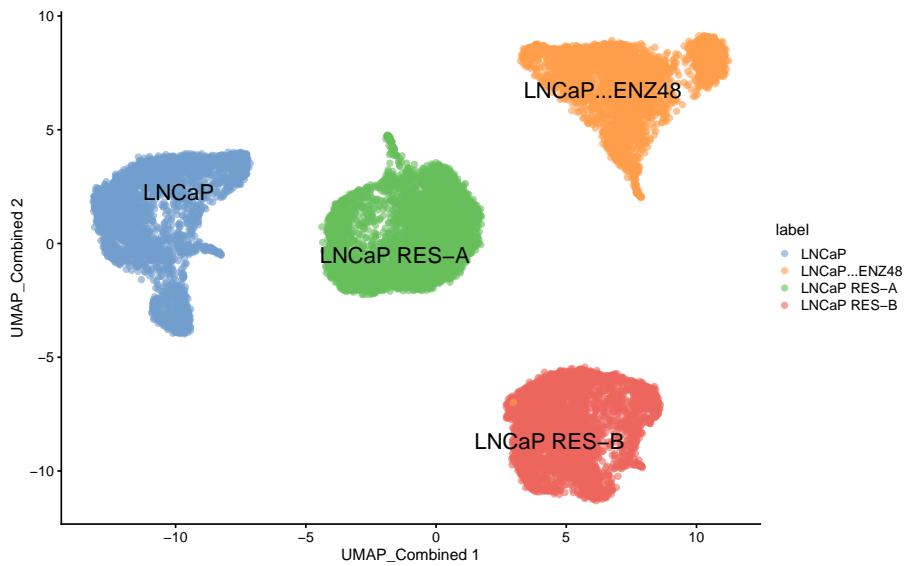
```
GeneExpressionMatrix <- as(GeneExpressionMatrix, "SingleCellExperiment")  
assayNames(GeneExpressionMatrix) <- "logcounts"  
assayNames(PeakMatrix) <- "counts"
```

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

```
reducedDim(GeneExpressionMatrix, "UMAP_Combined") <- getEmbedding(ArchRProj = proj,  
                                                               embedding = "UMAP_Combined",  
                                                               returnDF = TRUE)[colnames(GeneExpressionMatrix)]  
colData(GeneExpressionMatrix) <- getCellColData(proj)[colnames(GeneExpressionMatrix),]  
rownames(GeneExpressionMatrix) <- rowData(GeneExpressionMatrix)$name  
  
# add cell label  
GeneExpressionMatrix$label <- GeneExpressionMatrix$Cells  
GeneExpressionMatrix$label[GeneExpressionMatrix$Treatment == "enzalutamide 48h"] <- "LNCaP-ENZ48"  
GeneExpressionMatrix$label <- factor(GeneExpressionMatrix$label,  
                                      levels = c("LNCaP", "LNCaP-ENZ48", "LNCaP RES-A", "LNCaP RES-B"))
```

Visualize singleCellExperiment by UMAP

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



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")
#> see ?scMultiome and browseVignettes('scMultiome') for documentation
#> loading from cache
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 the `addPeak2GeneLinks` function from the ArchR package. The user would need to supply a path to an ArchR project already containing peak and gene matrices, as well as Latent semantic indexing (LSI) dimensionality reduction.

```
# path to ArchR project
p2g <- calculateP2G(ArchR_path = archR_project_path,
                      useDim = "iLSI_Combined",
                      useMatrix = "GeneIntegrationMatrix",
                      threads = 1)
#> Setting ArchRLogging = FALSE
#> Using ArchR to compute peak to gene links...
#> 2023-05-23 12:33:45.051952 : Getting Available Matrices, 0 mins elapsed.
#> 2023-05-23 12:33:49.075071 : Filtered Low Prediction Score Cells (0 of 15522, 0), 0.003 mins elapsed.
#> 2023-05-23 12:33:49.88252 : Computing KNN, 0.016 mins elapsed.
#> 2023-05-23 12:33:51.361712 : Identifying Non-Overlapping KNN pairs, 0.041 mins elapsed.
#> 2023-05-23 12:33:53.697356 : Identified 497 Groupings!, 0.08 mins elapsed.
#> 2023-05-23 12:33:53.751488 : Getting Group RNA Matrix, 0.081 mins elapsed.
#> 2023-05-23 12:36:07.326444 : Getting Group ATAC Matrix, 2.307 mins elapsed.
#> 2023-05-23 12:38:02.526708 : Normalizing Group Matrices, 4.227 mins elapsed.
#> 2023-05-23 12:38:10.283445 : Finding Peak Gene Pairings, 4.356 mins elapsed.
#> 2023-05-23 12:38:10.914075 : Computing Correlations, 4.367 mins elapsed.
#> 2023-05-23 12:38:22.705483 : Completed Peak2Gene Correlations!, 4.563 mins elapsed.
head(p2g)
#> DataFrame with 6 rows and 8 columns
#>   idxATAC     chr    start      end    idxRNA     target Correlation
#>   <integer> <factor> <integer> <integer> <integer> <character>  <numeric>
#> 1      15    chr1    912762    913262      7     NOC2L  0.543478
#> 2      25    chr1    920261    920761      7     NOC2L  0.602633
#> 3      25    chr1    920261    920761      8    KLHL17  0.597510
#> 4      32    chr1    927728    928228      7     NOC2L  0.660584
#> 5      32    chr1    927728    928228      8    KLHL17  0.532240
#> 6      73    chr1   1079335   1079835     23      SDF4  0.598232
#>   distance
#>   <numeric>
#> 1      46297
#> 2      38798
#> 3      40076
#> 4      31331
#> 5      32609
#> 6      152446
```

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

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)
#> DataFrame with 6 rows and 10 columns
#>   idxATAC     chr    start      end    idxRNA     target     corr
#>   <integer> <factor> <integer> <integer> <integer> <character> <matrix>
#> 1      15     chr1  912762  913262      7    NOC2L 0.543478
#> 2      15     chr1  912762  913262      7    NOC2L 0.543478
#> 3      15     chr1  912762  913262      7    NOC2L 0.543478
#> 4      15     chr1  912762  913262      7    NOC2L 0.543478
#> 5      15     chr1  912762  913262      7    NOC2L 0.543478
#> 6      15     chr1  912762  913262      7    NOC2L 0.543478
#>   distance     idxTF        tf
#>   <numeric> <integer> <character>
#> 1      46297       10     AG01
#> 2      46297       22    AML1-ETO
#> 3      46297       32    ARID4A
#> 4      46297       33    ARID4B
#> 5      46297       34    ARID5B
#> 6      46297       80     BCOR
```

Epiregulon computes weights using either correlation, linear regression, mutual information, log fold change or wilcoxon rank sum test. The choice of methods depends on the datasets. Correlation works best when increased TF activity results from increased TF expression, such in the case of normal development. The user has a choice between computing the correlation of TF expression vs target gene expression by setting `method = "corr"`, or the product of TF expression and chromatin accessibility at TF-bound regulatory elements vs target gene expression by setting `method = "corr"` and `"tf_re.merge = TRUE"`.

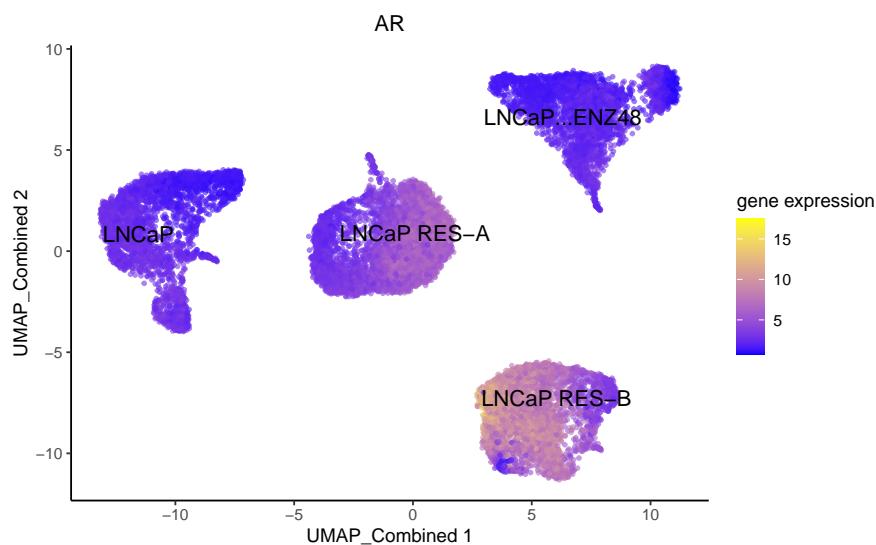
In the case of drug treatment, however, the activity of TF is suppressed often not by downregulation of the TF gene expression, but by direct interference of the TF protein function. In this dataset, the drug enzalutamide blocks the ligand binding domain of the

prostate cancer archr tutorial

androgen receptor and prevents it from binding to the chromatin. As a result, while the AR gene expression stays the same, the chromatin accessibility of AR, as computed by chromVar in the ArchR package, is greatly reduced by 48 hour treatment of enzalutamide.

First, we visualize the AR expression and observed that enzalutamide did not decrease AR expression.

```
plotActivityDim(sce = GeneExpressionMatrix,
                 activity_matrix = assay(GeneExpressionMatrix),
                 tf = "AR",
                 dimtype = "UMAP_Combined",
                 label = "label",
                 point_size = 1,
                 legend.label = "gene expression")
```

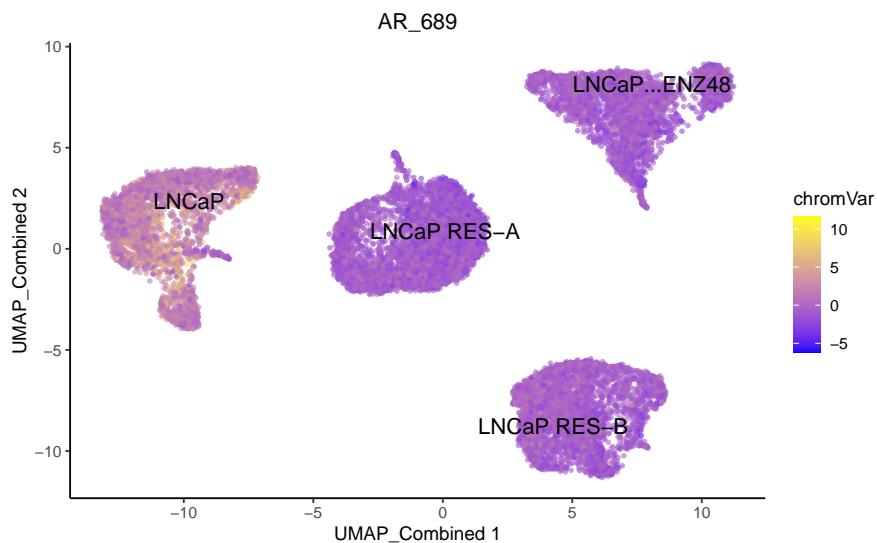


Then we extract the chromVarMatrix from ArchR project and then visualize the chromatin accessibility at AR bound sites. We can see that 48 hour of enzalutamide treatment reduced chromatin accessibility at AR bound sites

```
chromVarMatrix <- getMatrixFromProject(
  ArchRProj = proj,
  useMatrix = "MotifMatrix",
  useSeqnames = NULL,
  verbose = TRUE,
  binarize = FALSE,
  threads = 1
)
#> 2023-05-23 12:39:49.631269 : Organizing colData, 0.323 mins elapsed.
#> 2023-05-23 12:39:49.761175 : Organizing rowData, 0.325 mins elapsed.
#> 2023-05-23 12:39:49.76327 : Organizing rowRanges, 0.325 mins elapsed.
#> 2023-05-23 12:39:49.767578 : Organizing Assays (1 of 2), 0.325 mins elapsed.
#> 2023-05-23 12:39:50.282168 : Organizing Assays (2 of 2), 0.334 mins elapsed.
#> 2023-05-23 12:39:50.797529 : Constructing SummarizedExperiment, 0.343 mins elapsed.
#> 2023-05-23 12:39:51.948802 : Finished Matrix Creation, 0.362 mins elapsed.
```

prostate cancer archr tutorial

```
plotActivityDim(sce = GeneExpressionMatrix,
                 activity_matrix = assay(chromVarMatrix, "z"),
                 tf = "AR_689",
                 dimtype = "UMAP_Combined",
                 label = "label",
                 point_size = 1,
                 legend.label = "chromVar")
```



Therefore, we consider the choice of the `wilcoxon` test which compare target gene expression in cells meeting both the TF expression and accessibility cutoffs vs cells failing either the TF expression or/and accessibility cutoffs. We also compare the output of `wilcoxon` vs `corr`.

```
regulon.w.wilcox <- addWeights(regulon = pruned.regulon,
                                 expMatrix = GeneExpressionMatrix,
                                 exp_assay = "logcounts",
                                 peakMatrix = PeakMatrix,
                                 peak_assay = "counts",
                                 clusters = GeneExpressionMatrix$Sample,
                                 method = "wilcoxon")
#> adding weights using wilcoxon...
regulon.w.corr <- addWeights(regulon = pruned.regulon,
                               expMatrix = GeneExpressionMatrix,
                               exp_assay = "logcounts",
                               peakMatrix = PeakMatrix,
                               peak_assay = "counts",
                               clusters = GeneExpressionMatrix$Sample,
                               method = "corr")
#> adding weights using corr...
#> calculating average expression across clusters...
#> computing weights...
regulon.w.corr.re <- addWeights(regulon = pruned.regulon,
```

```

expMatrix = GeneExpressionMatrix,
exp_assay = "logcounts",
peakMatrix = PeakMatrix,
peak_assay = "counts",
clusters = GeneExpressionMatrix$Sample,
method = "corr",
tf_re.merge = TRUE)

#> adding weights using corr...
#> calculating average expression across clusters...
#> computing weights...

```

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.

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

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

We calculate three different activities corresponding to the different weighted regulons

```

score.combine.wilcox <- calculateActivity(expMatrix = GeneExpressionMatrix,
                                             regulon = regulon.w.wilcox,
                                             normalize = TRUE,
                                             mode = "weight",
                                             method = "weightedMean")

#> calculating TF activity from regulon using weightedmean
#> aggregating regulons...
#> creating weight matrix...
#> calculating activity scores...
#> normalize by mean...
#> normalize by the number of targets...

score.combine.corr <- calculateActivity(expMatrix = GeneExpressionMatrix,
                                           regulon = regulon.w.corr,
                                           normalize = TRUE,
                                           mode = "weight",
                                           method = "weightedMean")

#> calculating TF activity from regulon using weightedmean
#> aggregating regulons...
#> creating weight matrix...
#> calculating activity scores...
#> normalize by mean...
#> normalize by the number of targets...

score.combine.corr.re <- calculateActivity(expMatrix = GeneExpressionMatrix,
                                              regulon = regulon.w.corr.re,
                                              normalize = TRUE,
                                              mode = "weight",
                                             

```

prostate cancer archr tutorial

```
method = "weightedMean")  
#> calculating TF activity from regulon using weightedmean  
#> aggregating regulons...  
#> creating weight matrix...  
#> calculating activity scores...  
#> normalize by mean...  
#> normalize by the number of targets...
```

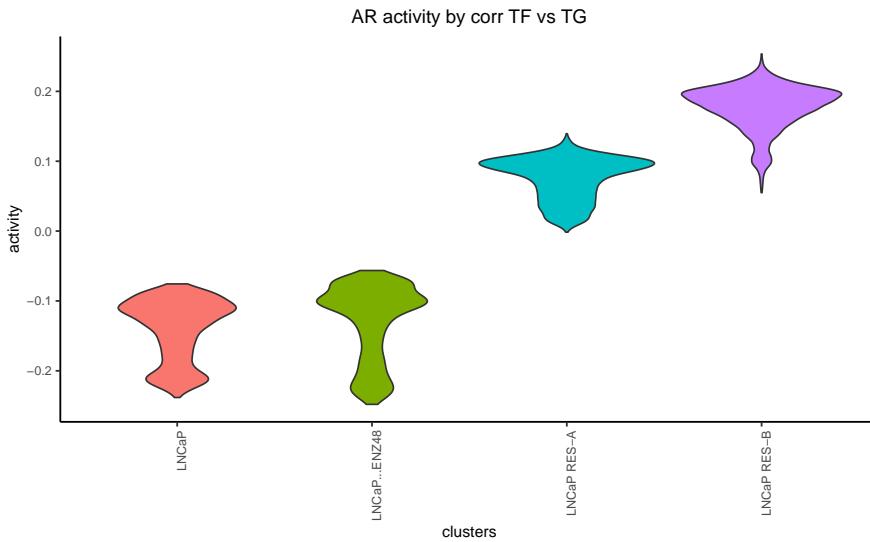
We visualize the different activities side by side

```
plotActivityViolin(activity_matrix = score.combine.wilcox,  
tf = c( "AR"),  
clusters = GeneExpressionMatrix$label) + ggtitle ("AR activity by wilcoxon")
```

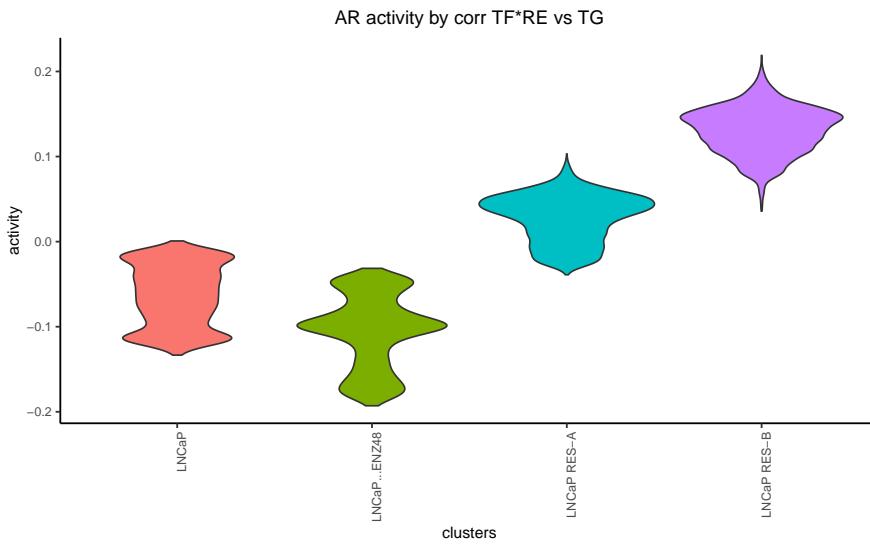


```
plotActivityViolin(activity_matrix = score.combine.corr,  
tf = c( "AR"),  
clusters = GeneExpressionMatrix$label) + ggtitle ("AR activity by corr TF vs TG")
```

prostate cancer archr tutorial



```
plotActivityViolin(activity_matrix = score.combine.corr.re,
                    tf = c( "AR"),
                    clusters = GeneExpressionMatrix$label) + ggtitle ("AR activity by corr TF*RE vs TG")
```



4.6 Perform differential activity

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

prostate cancer archr tutorial

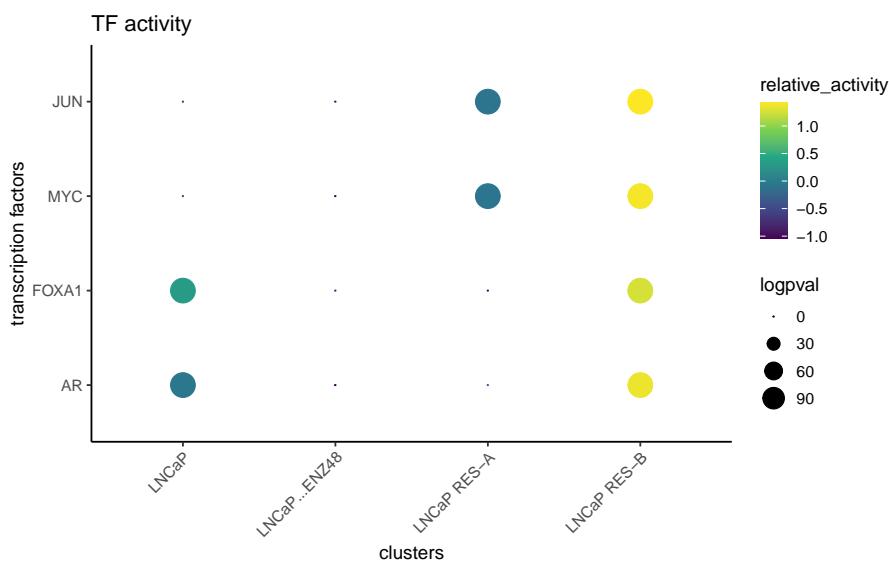
Take the top TFs

```
markers.sig <- getSigGenes(markers, topgenes = 8 )
#> Using a logFC cutoff of 0 for class LNCaP
#> Using a logFC cutoff of 0 for class LNCaP-ENZ48
#> Using a logFC cutoff of 0 for class LNCaP RES-A
#> Using a logFC cutoff of 0 for class LNCaP RES-B
```

4.7 Visualize the results

First visualize the known differential TFs by bubble plot

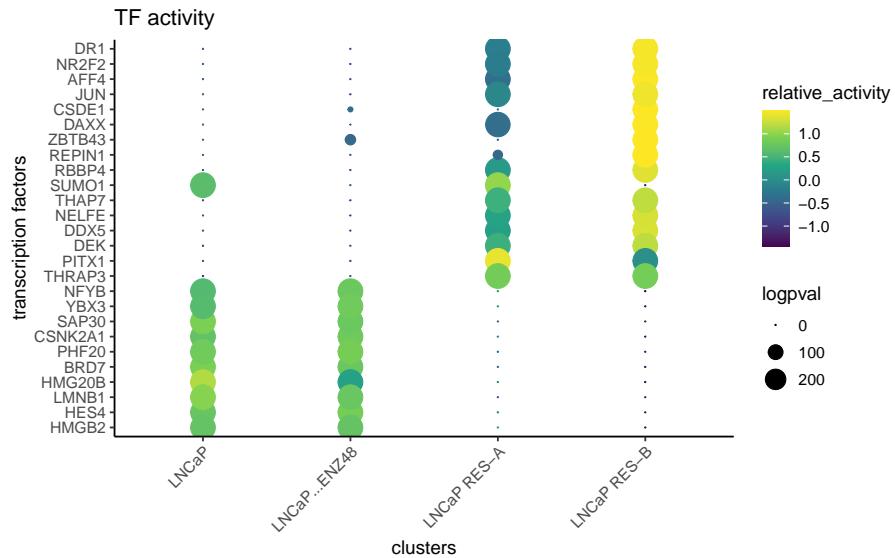
```
plotBubble(activity_matrix = score.combine.wilcox,
           tf = c("AR", "FOXA1", "MYC", "JUN"),
           clusters = GeneExpressionMatrix$label)
```



Then visualize the most differential TFs by clusters

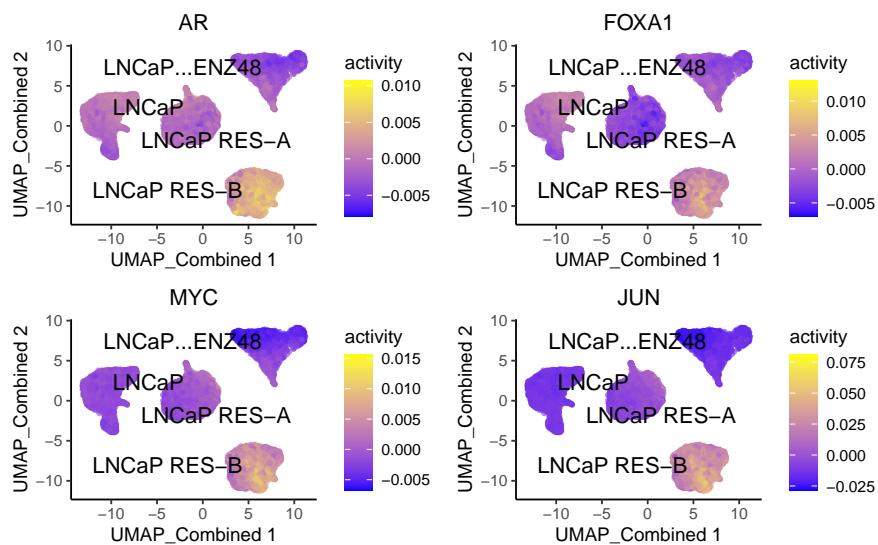
```
plotBubble(activity_matrix = score.combine.wilcox,
           tf = markers.sig$tf,
           clusters = GeneExpressionMatrix$label)
```

prostate cancer archr tutorial



Visualize the known differential TFs by UMAP

```
plotActivityDim(sce = GeneExpressionMatrix,
                 activity_matrix = score.combine.wilcox,
                 tf = c( "AR", "FOXA1", "MYC", "JUN"),
                 dimtype = "UMAP_Combined",
                 label = "label",
                 point_size = 1,
                 ncol = 2,
                 nrow = 2)
```

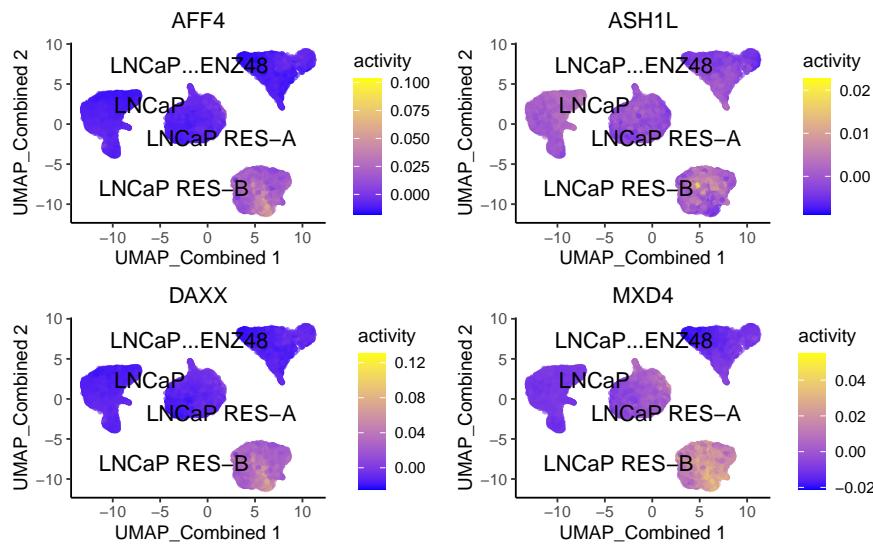


Visualize the newly discovered differential TFs by UMAP

```
plotActivityDim(sce = GeneExpressionMatrix,
                 activity_matrix = score.combine.wilcox,
```

prostate cancer archr tutorial

```
tf = c("AFF4", "ASH1L", "DAXX", "MXD4"),
dimtype = "UMAP_Combined",
label = "label",
point_size = 1,
ncol = 2,
nrow = 2)
```

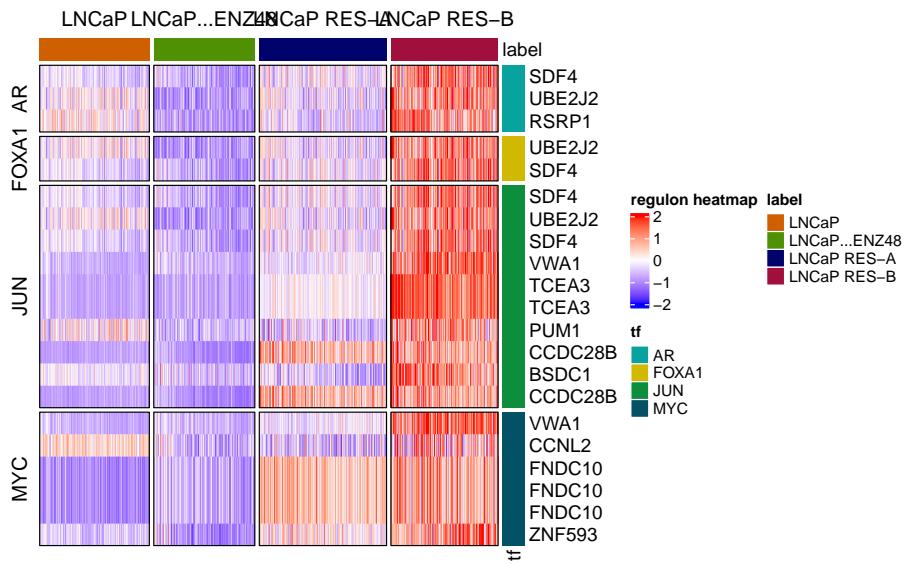


Visualize regulons by heatmap

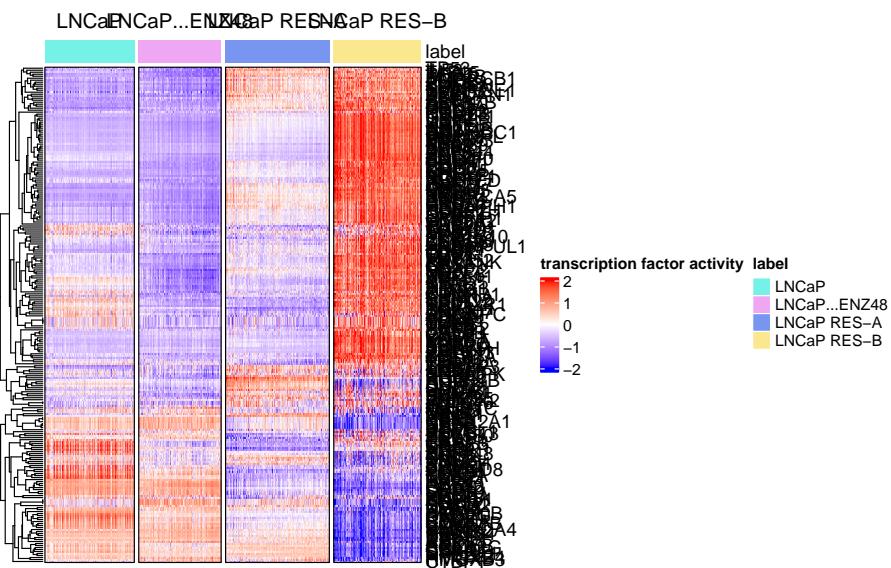
```
rowData(GeneExpressionMatrix) <- NULL

plotHeatmapRegulon(sce=GeneExpressionMatrix,
                    tfs= c( "AR", "FOXA1", "MYC", "JUN"),
                    regulon=regulon.w.wilcox,
                    regulon_cutoff=0.1,
                    downsample=1000,
                    cell_attributes="label",
                    col_gap="label",
                    exprs_values="logcounts",
                    name="regulon heatmap")
```

prostate cancer archr tutorial



```
plotHeatmapActivity(activity=score.combine.wilcox,
                     sce=GeneExpressionMatrix,
                     tfs=rownames(score.combine.wilcox),
                     downsample=1000,
                     cell_attributes="label",
                     col_gap="label",
                     name = "transcription factor activity")
```



4.8 Geneset enrichment

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

prostate cancer archr tutorial

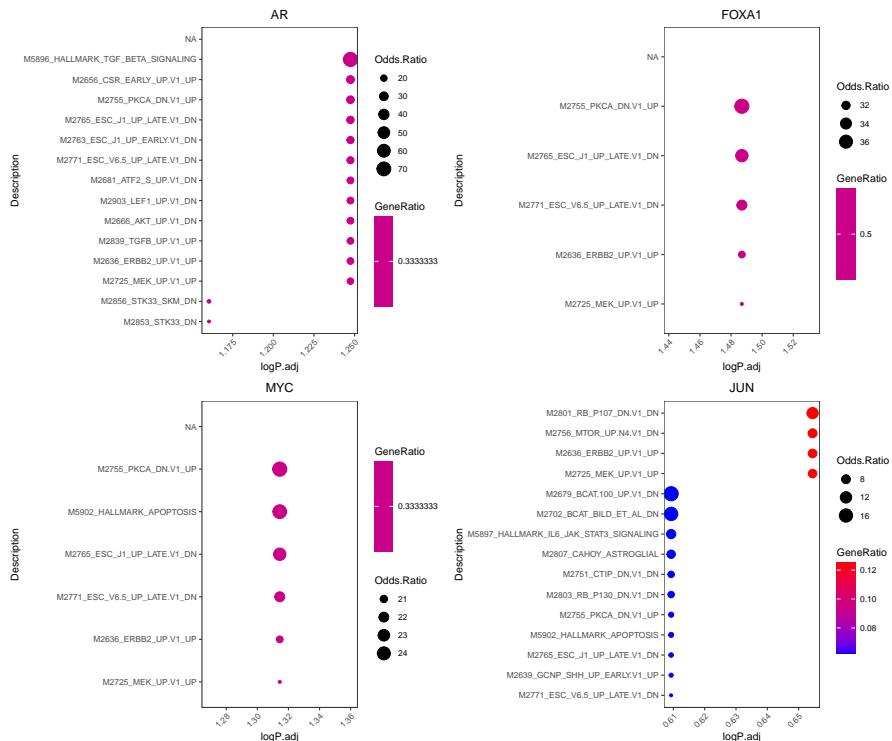
```
#retrieve genesets
H <- EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb",
                                      cat = "H", gene.id.type = "SYMBOL",
                                      cache = FALSE)
C6 <- EnrichmentBrowser::getGenesets(org = "hsa", db = "msigdb",
                     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

#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("AR", "FOXA1", "MYC", "JUN"),
                                 regulon = regulon.w.wilcox,
                                 weight = "weight",
                                 weight_cutoff = 0.1,
                                 genesets = gs.list)

#> AR
#>
#> FOXA1
#> MYC
#> JUN

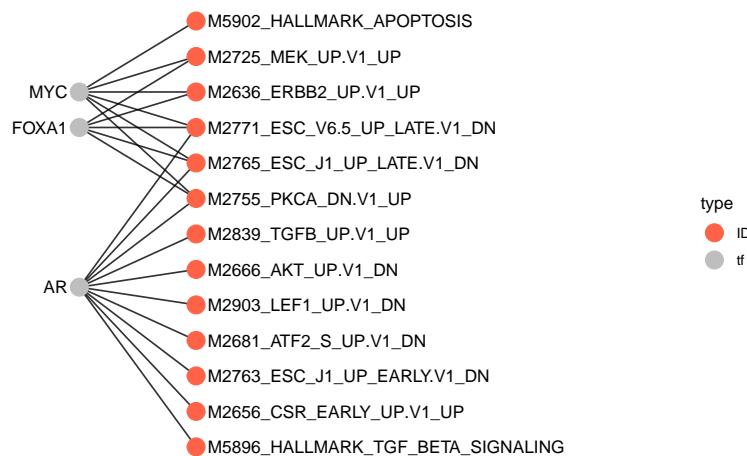
#plot results
enrichPlot(results = enrichresults, ncol = 2)
```



4.9 Network analysis

We can visualize the genesets as a network

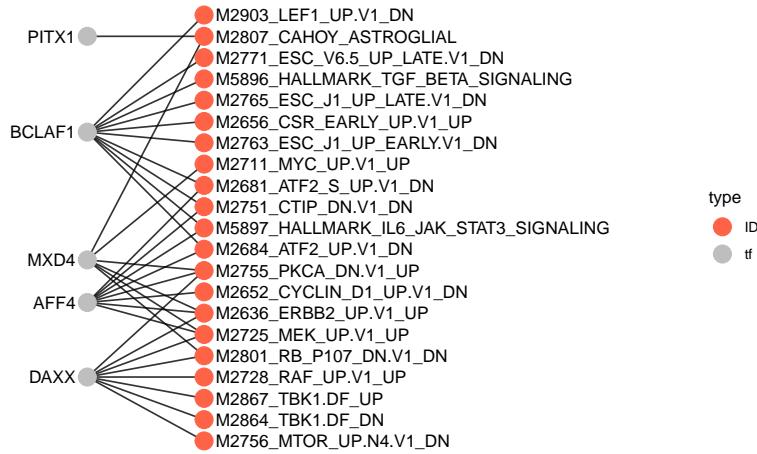
```
plotGseaNetwork(tf = names(enrichresults),
                enrichresults = enrichresults,
                p.adj_cutoff = 0.1,
                ntop_pathways = 10)
```



```
enrichresults <- regulonEnrich(TF = c("AFF4", "ASH1L", "DAXX", "MDX4", "CDC73", "REPIN1",
                                         "BCLAF1", "RBM34", "CERS6", "TOP2B", "ATF5", "PITX1",
                                         "EWSR1", "TOP2A"),
                                 regulon = regulon.w.wilcox,
                                 weight = "weight",
                                 weight_cutoff = 0.1,
                                 genesets = gs.list)

#> AFF4
#> ASH1L
#> DAXX
#> MDX4
#> CDC73
#> REPIN1
#> BCLAF1
#> RBM34
#> CERS6
#> TOP2B
#> ATF5
#> PITX1
#> EWSR1
#> TOP2A
plotGseaNetwork(tf = names(enrichresults),
                enrichresults = enrichresults,
                p.adj_cutoff = 0.1,
```

```
ntop_pathways = 10)
```



5 Session Info

```
sessionInfo()
#> R Under development (unstable) (2022-11-21 r83371)
#> 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
#> [4] LC_COLLATE=C LC_MONETARY=C LC_MESSAGES=C
#> [7] LC_PAPER=C LC_NAME=C LC_ADDRESS=C
#> [10] LC_TELEPHONE=C LC_MEASUREMENT=C LC_IDENTIFICATION=C
#>
#> time zone: Etc/UTC
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] grid      stats4     stats      graphics   grDevices  utils      datasets 
#> [8] methods    base      
#>
#> other attached packages:
#> [1] org.Hs.eg.db_3.17.0      AnnotationDbi_1.62.0
#> [3] msigdbr_7.5.1            nabor_0.5.0
```

prostate cancer archr tutorial

```
#> [5] scMultiome_1.1.0           MultiAssayExperiment_1.25.11
#> [7] ExperimentHub_2.7.1       AnnotationHub_3.7.4
#> [9] BiocFileCache_2.8.0       dbplyr_2.3.2
#> [11] rhdf5_2.44.0             RcppArmadillo_0.12.2.0.0
#> [13] Rcpp_1.0.10              Matrix_1.5-3
#> [15] sparseMatrixStats_1.11.1 data.table_1.14.8
#> [17] stringr_1.5.0            plyr_1.8.8
#> [19] magrittr_2.0.3           ggplot2_3.4.2
#> [21] gtable_0.3.3             gtools_3.9.4
#> [23] gridExtra_2.3            devtools_2.4.5
#> [25] usethis_2.1.6            ArchR_1.0.3
#> [27] epiregulon_1.0.23        SingleCellExperiment_1.21.1
#> [29] SummarizedExperiment_1.29.1 Biobase_2.60.0
#> [31] GenomicRanges_1.52.0      GenomeInfoDb_1.36.0
#> [33] IRanges_2.34.0           S4Vectors_0.38.0
#> [35] BiocGenerics_0.46.0       MatrixGenerics_1.12.0
#> [37] matrixStats_0.63.0        BiocStyle_2.27.2
#>
#> loaded via a namespace (and not attached):
#> [1] fs_1.6.2                  GSVA_1.47.3
#> [3] bitops_1.0-7              enrichplot_1.19.2
#> [5] HDO.db_0.99.1             httr_1.4.5
#> [7] RColorBrewer_1.1-3        doParallel_1.0.17
#> [9] Rgraphviz_2.43.0          profvis_0.3.7
#> [11] tools_4.3.0              backports_1.4.1
#> [13] utf8_1.2.3               R6_2.5.1
#> [15] HDF5Array_1.28.1         lazyeval_0.2.2
#> [17] rhdf5filters_1.11.2      GetoptLong_1.0.5
#> [19] urlchecker_1.0.1         withr_2.5.0
#> [21] prettyunits_1.1.1        cli_3.6.1
#> [23] Cairo_1.6-0              scatterpie_0.1.9
#> [25] labeling_0.4.2           KEGGgraph_1.59.3
#> [27] yulab.utils_0.0.6        gson_0.1.0
#> [29] DOSE_3.25.0              scater_1.27.9
#> [31] sessioninfo_1.2.2        limma_3.55.10
#> [33] rstudioapi_0.14          RSQLite_2.3.1
#> [35] gridGraphics_0.5-1        generics_0.1.3
#> [37] shape_1.4.6              dplyr_1.1.2
#> [39] GO.db_3.17.0             ggbeeswarm_0.7.1
#> [41] fansi_1.0.4              lifecycle_1.0.3
#> [43] yaml_2.3.7               edgeR_3.41.9
#> [45] qvalue_2.31.1            blob_1.2.4
#> [47] promises_1.2.0.1          dqrng_0.3.0
#> [49] crayon_1.5.2             miniUI_0.1.1.1
#> [51] lattice_0.20-45           beachmat_2.15.2
#> [53] cowplot_1.1.1            annotate_1.77.0
#> [55] KEGGREST_1.40.0           magick_2.7.4
#> [57] pillar_1.9.0              knitr_1.42
#> [59] ComplexHeatmap_2.15.3     metapod_1.7.0
#> [61] fgsea_1.25.2              rjson_0.2.21
#> [63] codetools_0.2-18           fastmatch_1.1-3
```

prostate cancer archr tutorial

```
#> [65] glue_1.6.2                      ggfun_0.0.9
#> [67] downloader_0.4                   remotes_2.4.2
#> [69] treeio_1.23.1                  vctrs_0.6.2
#> [71] png_0.1-8                     cachem_1.0.8
#> [73] xfun_0.39                    S4Arrays_1.0.1
#> [75] mime_0.12                     tidygraph_1.2.3
#> [77] iterators_1.0.14                statmod_1.5.0
#> [79] bluster_1.9.1                 interactiveDisplayBase_1.37.0
#> [81] ellipsis_0.3.2                nlme_3.1-162
#> [83] ggtree_3.7.2                  bit64_4.0.5
#> [85] filelock_1.0.2                irlba_2.3.5.1
#> [87] viper_0.4.5                  colorspace_2.1-0
#> [89] DBI_1.1.3                    tidyselect_1.2.0
#> [91] processx_3.8.0                bit_4.0.5
#> [93] compiler_4.3.0                curl_5.0.0
#> [95] graph_1.77.3                 BiocNeighbors_1.17.1
#> [97] DelayedArray_0.26.2            shadowtext_0.1.2
#> [99] bookdown_0.33                 checkmate_2.2.0
#> [101] scales_1.2.1                 callr_3.7.3
#> [103] rappdirs_0.3.3                digest_0.6.31
#> [105] rmarkdown_2.21                XVector_0.40.0
#> [107] htmltools_0.5.5              pkgconfig_2.0.3
#> [109] fastmap_1.1.1                rlang_1.1.1
#> [111] GlobalOptions_0.1.2            htmlwidgets_1.6.2
#> [113] shiny_1.7.4                  DelayedMatrixStats_1.21.0
#> [115] farver_2.1.1                 jsonlite_1.8.4
#> [117] BiocParallel_1.34.0            GOSemSim_2.25.0
#> [119] BiocSingular_1.15.0            RCurl_1.98-1.12
#> [121] ggplotify_0.1.0               scuttle_1.9.4
#> [123] GenomeInfoDbData_1.2.10        patchwork_1.1.2
#> [125] Rhdf5lib_1.21.1              munsell_0.5.0
#> [127] ape_5.7-1                   babelgene_22.9
#> [129] viridis_0.6.3                EnrichmentBrowser_2.29.2
#> [131] stringi_1.7.12              ggraph_2.1.0
#> [133] MASS_7.3-58.1               zlibbioc_1.46.0
#> [135] pkgbuild_1.4.0                parallel_4.3.0
#> [137] ggrepel_0.9.3               graphlayouts_0.8.4
#> [139] Biostrings_2.68.0             splines_4.3.0
#> [141] circlize_0.4.15            locfit_1.5-9.7
#> [143] ps_1.7.2                   igraph_1.4.2
#> [145] reshape2_1.4.4              ScaledMatrix_1.7.1
#> [147] pkgload_1.3.2              BiocVersion_3.17.1
#> [149] XML_3.99-0.14              evaluate_0.20
#> [151] scran_1.27.3              BiocManager_1.30.20
#> [153] tweenr_2.0.2              foreach_1.5.2
#> [155] httpuv_1.6.9              polyclip_1.10-4
#> [157] tidyverse_1.3.0             purrr_1.0.1
#> [159] clue_0.3-64                ggforce_0.4.1
#> [161] rsvd_1.0.5                xtable_1.8-4
#> [163] tidytree_0.4.2             later_1.3.0
#> [165] viridisLite_0.4.2            tibble_3.2.1
```

prostate cancer archr tutorial

```
#> [167] aplot_0.1.10          clusterProfiler_4.9.0
#> [169] memoise_2.0.1         beeswarm_0.4.0
#> [171] cluster_2.1.4         GSEABase_1.61.2
```