

perform gene ontology analysis on scRNA-seq data

Qingnan Liang

8/21/2022

Introduction

Here is a demonstration of how to use the package ‘gsdensity’ to perform gene set analysis on single-cell RNA-seq data. Mostly, the analysis of scRNA-seq data is ‘cell centric’, which first identifies groups of cells which are somehow interesting, and then find out the specialty/function of them. However, in many cases, we do not have much anticipation of which cells we should go after, especially when the cell type/state annotation is yet unclear, and there is no systematic methods to nominate cells of interest (largely depends on the experiences of the researcher). Instead, experts in the very field may have in mind some pathways relevant to this sample (“Dr.XXX, could you please take a look at the XXX pathway in your data?”).

The motivation of this package is to allow for ‘pathway centric’ analysis of single-cell data. We try to answer two questions:

First, given a pathway (in the format of a set of genes) and a cell-by-gene matrix, can we tell if the gene set is relatively enriched by a subset cells or not (without any clustering/annotation of the cells)?

Second, if a pathway is relatively enriched by some cells, can we fetch these cells?

The following tutorial will give an example using the pbmc3k data and gene ontology (GO) biological process gene sets.

Load libraries

```
library(gsdensity)
library(ggplot2)  # for plotting
library(reshape2)
library(msigdb)   # for gathering gene sets
library(Seurat)
library(SeuratData)
library(future)   # for parallel computing
library(future.apply) # for parallel computing
```

Preparation: Collect gene sets

Gene sets will be used as inputs of the analysis. The format of gene sets is ‘list’ in R.

```
gene.set.manual <- list(gene.set.1 = c("gene_a", "gene_b", "gene_c"),
  gene.set.2 = c("gene_d", "gene_e", "gene_f"), gene.set.2 = c("gene_g",
    "gene_h", "gene_i"))
gene.set.manual
```

1. We can create gene sets manually:

```
## $gene.set.1
## [1] "gene_a" "gene_b" "gene_c"
##
## $gene.set.2
## [1] "gene_d" "gene_e" "gene_f"
##
## $gene.set.2
## [1] "gene_g" "gene_h" "gene_i"
```

```
# Collect a single category from the msigdb database For
# more information please check:
# https://cran.r-project.org/web/packages/msigdb/vignettes/msigdb-intro.html

mdb_c5 <- msigdb(species = "Homo sapiens", category = "C5")

# If we just want to do biological process:
mdb_c5_bp <- mdb_c5[mdb_c5$gs_subcat == "GO:BP", ]

# convert msigdb gene sets to a list good for the input
gene.set.list <- list()
for (gene.set.name in unique(mdb_c5_bp$gs_name)) {
  gene.set.list[[gene.set.name]] <- mdb_c5_bp[mdb_c5_bp$gs_name %in%
    gene.set.name, ]$gene_symbol
}
str(gene.set.list[1:10])
```

2. We can gather gene sets from public databases. Here I show examples using msigdb which has common gene sets including GOs, Hallmark genes:

```
## List of 10
## $ GOBP_10_FORMYLTETRAHYDROFOLATE_METABOLIC_PROCESS : chr [1:6] "AASDHPPT" "ALDH1L1" "ALDH1L2"
## $ GOBP_2_OXOGLUTARATE_METABOLIC_PROCESS : chr [1:18] "AADAT" "ADHFE1" "D2HGDH"
## $ GOBP_2FE_2S_CLUSTER_ASSEMBLY : chr [1:6] "BOLA2" "BOLA2B" "GLRX3"
## $ GOBP_3_PHOSPHOADENOSINE_5_PHOSPHOSULFATE_BIOSYNTHETIC_PROCESS : chr [1:6] "PAPSS1" "PAPSS2" "SLC26A3"
## $ GOBP_3_PHOSPHOADENOSINE_5_PHOSPHOSULFATE_METABOLIC_PROCESS : chr [1:21] "ABHD14B" "BPNT1" "ENPP1"
## $ GOBP_3_UTR_MEDIATED_MRNA_DESTABILIZATION : chr [1:19] "CPEB3" "DHX36" "DHX36"
## $ GOBP_3_UTR_MEDIATED_MRNA_STABILIZATION : chr [1:24] "ANGEL2" "BOLL" "DAZ1"
## $ GOBP_5_PHOSPHORIBOSE_1_DIPHOSPHATE_METABOLIC_PROCESS : chr [1:6] "PRPS1" "PRPS1L1" "PRPS2"
## $ GOBP_5S_CLASS_RRNA_TRANSCRIPTION_BY_RNA_POLYMERASE_III : chr [1:6] "GTF3C1" "GTF3C2" "GTF3C3"
## $ GOBP_ABSCISSION : chr [1:6] "AURKB" "CHMP4C" "IST1"
```

```

# for the purpose of getting a record, we can create and
# save a dataframe which has the name of gene sets and the
# members of it.
genes <- sapply(gene.set.list, function(x) paste(x, collapse = ", "))
gene.set.list.df <- cbind(gene.set = names(gene.set.list), genes = genes)
rownames(gene.set.list.df) <- 1:nrow(gene.set.list.df)
head(gene.set.list.df)

```

```

##   gene.set
## 1 "GOBP_10_FORMYLTETRAHYDROFOLATE_METABOLIC_PROCESS"
## 2 "GOBP_2_OXOGLUTARATE_METABOLIC_PROCESS"
## 3 "GOBP_2FE_2S_CLUSTER_ASSEMBLY"
## 4 "GOBP_3_PHOSPHOADENOSINE_5_PHOSPHOSULFATE_BIOSYNTHETIC_PROCESS"
## 5 "GOBP_3_PHOSPHOADENOSINE_5_PHOSPHOSULFATE_METABOLIC_PROCESS"
## 6 "GOBP_3_UTR_MEDIATED_MRNA_DESTABILIZATION"
##   genes
## 1 "AASDHPPT, ALDH1L1, ALDH1L2, MTHFD1, MTHFD1L, MTHFD2L"
## 2 "AADAT, ADHFE1, D2HGDH, DLST, GOT1, GOT2, GPT2, IDH1, IDH2, IDH3B, KYAT3, L2HGDH, MRPS36, MRPS36, "
## 3 "BOLA2, BOLA2B, GLRX3, GLRX5, HSCB, NFS1"
## 4 "PAPSS1, PAPSS2, SLC26A1, SLC26A2, SLC35B2, SLC35B3"
## 5 "ABHD14B, BPNT1, ENPP1, PAPSS1, PAPSS2, SLC26A1, SLC26A2, SLC35B2, SLC35B3, SULT1A1, SULT1A2, SULT1A3"
## 6 "CPEB3, DHX36, DHX36, DND1, DND1, HNRNPD, KHSRP, MOV10, PLEKHN1, RBM24, RC3H1, TARDBP, TRIM71, UPF1"

```

```

# can be output for record write.csv(gene.set.list.df,
# 'gene.set.list.df.csv')

```

Preparation: Single-cell datasets

```

data("pbmc3k")
print(pbmc3k)

```

Here we use the pbmc3k data from SeuratData

```

## An object of class Seurat
## 13714 features across 2700 samples within 1 assay
## Active assay: RNA (13714 features, 0 variable features)

```

```

# Normalize the data
pbmc3k <- NormalizeData(pbmc3k)

```

1. Find out gene sets being relatively enriched by some cell populations

```

# Compute cell/gene embeddings By default, the first 10
# dims of MCA space are used; can be adjusted by setting
# dims.use By default, all genes from the single cell data

```

```

# is used; alternatively, we can use only the genes
# appearing in the gene sets; a sufficient number of genes
# should be retained (a few thousands) for the purpose of
# calculating the background; can be adjusted by setting
# genes.use This step is using the excellent package
# 'CellID' https://github.com/RausellLab/CellID. This step
# may require more RAM (hundreds of Gbs) to run bigger
# datasets (several tens of thousands) because of the MCA
# calculation Alternatively, averaging similar cells will
# shrink the size of the dataset. Can check
# https://github.com/qingnanl/SRAVG or similar tools

ce <- compute.mca(object = pbmc3k)

```

We achieve this by using multiple correspondence analysis (MCA) to co-embed genes and cells; thus the gene coordinates in the space are affected by their relationship to cells. Basically we are trying to find out if the distribution of a gene set is very different from the background (all genes), and if so, it indicates that the gene set is displaying affinity towards some cells.

```

## 1.01 sec elapsed
## 75.35 sec elapsed
## 3.57 sec elapsed

```

```
head(ce)
```

```

##           mca_1      mca_2      mca_3      mca_4      mca_5
## AL627309.1  1.00025758 -0.09801091  0.2576701 -0.4123061 -0.01149743
## AP006222.2  0.28756691 -0.27481812 -0.2942241 -0.2271572  0.28836018
## RP11-206L10.2 0.02327183  0.08446649  0.2896421  1.4898457  0.10507789
## RP11-206L10.9 2.09918617 -0.52055412 -0.1097032  0.1316594  0.48677281
## LINC00115    0.18638658 -0.26210325 -0.2019776  0.1714087 -0.15072327
## NOC2L       -0.03550786 -0.17197694 -0.1020475  0.1442398  0.04186578
##           mca_6      mca_7      mca_8      mca_9      mca_10
## AL627309.1  -0.4478389 -0.75748020 -0.23269822 -0.30348777  0.05231671
## AP006222.2  0.1485448 -0.64504601 -0.02083979 -0.67381873  0.45130507
## RP11-206L10.2 0.1616784 -0.56138732 -0.52713575 -0.05021750  0.43872880
## RP11-206L10.9 0.1496896 -0.68845718 -0.40160759 -0.58984624  0.36267276
## LINC00115    0.1721549 -0.15439943  0.03849592 -0.08469765 -0.03258933
## NOC2L       0.1180863 -0.04785021 -0.03176059  0.05663379  0.02066340

```

```

# Compute the relative enrichment of each gene list

# It is strongly recommended to run this analysis with
# multi-threading, which speed up a lot Set 20 workers with
# the following line plan(multiprocess, workers = 20)

# For demonstration purpose, we want to only focus on 'B
# Cell' related gene sets (B cells are known to be included
# in the pbmc data)

gs.names <- grep("B_CELL", names(gene.set.list), value = T)
gs.names # 35 gene sets

```

```
## [1] "GOBP_B_1_B_CELL_DIFFERENTIATION"
## [2] "GOBP_B_CELL_ACTIVATION"
## [3] "GOBP_B_CELL_ACTIVATION_INVOLVED_IN_IMMUNE_RESPONSE"
## [4] "GOBP_B_CELL_APOPTOTIC_PROCESS"
## [5] "GOBP_B_CELL_CHEMOTAXIS"
## [6] "GOBP_B_CELL_DIFFERENTIATION"
## [7] "GOBP_B_CELL_HOMEOSTASIS"
## [8] "GOBP_B_CELL_LINEAGE_COMMITMENT"
## [9] "GOBP_B_CELL_MEDIATED_IMMUNITY"
## [10] "GOBP_B_CELL_PROLIFERATION"
## [11] "GOBP_B_CELL_PROLIFERATION_INVOLVED_IN_IMMUNE_RESPONSE"
## [12] "GOBP_B_CELL_RECEPTOR_SIGNALING_PATHWAY"
## [13] "GOBP_GERMINAL_CENTER_B_CELL_DIFFERENTIATION"
## [14] "GOBP_IMMATURE_B_CELL_DIFFERENTIATION"
## [15] "GOBP_MARGINAL_ZONE_B_CELL_DIFFERENTIATION"
## [16] "GOBP_MATURE_B_CELL_DIFFERENTIATION"
## [17] "GOBP_NEGATIVE_REGULATION_OF_B_CELL_ACTIVATION"
## [18] "GOBP_NEGATIVE_REGULATION_OF_B_CELL_APOPTOTIC_PROCESS"
## [19] "GOBP_NEGATIVE_REGULATION_OF_B_CELL_DIFFERENTIATION"
## [20] "GOBP_NEGATIVE_REGULATION_OF_B_CELL_MEDIATED_IMMUNITY"
## [21] "GOBP_NEGATIVE_REGULATION_OF_B_CELL_PROLIFERATION"
## [22] "GOBP_NEGATIVE_REGULATION_OF_B_CELL_RECEPTOR_SIGNALING_PATHWAY"
## [23] "GOBP_POSITIVE_REGULATION_OF_B_CELL_ACTIVATION"
## [24] "GOBP_POSITIVE_REGULATION_OF_B_CELL_DIFFERENTIATION"
## [25] "GOBP_POSITIVE_REGULATION_OF_B_CELL_MEDIATED_IMMUNITY"
## [26] "GOBP_POSITIVE_REGULATION_OF_B_CELL_PROLIFERATION"
## [27] "GOBP_POSITIVE_REGULATION_OF_B_CELL_RECEPTOR_SIGNALING_PATHWAY"
## [28] "GOBP_PRO_B_CELL_DIFFERENTIATION"
## [29] "GOBP_REGULATION_OF_B_CELL_ACTIVATION"
## [30] "GOBP_REGULATION_OF_B_CELL_APOPTOTIC_PROCESS"
## [31] "GOBP_REGULATION_OF_B_CELL_DIFFERENTIATION"
## [32] "GOBP_REGULATION_OF_B_CELL_MEDIATED_IMMUNITY"
## [33] "GOBP_REGULATION_OF_B_CELL_PROLIFERATION"
## [34] "GOBP_REGULATION_OF_B_CELL_RECEPTOR_SIGNALING_PATHWAY"
## [35] "GOBP_REGULATION_OF_PRO_B_CELL_DIFFERENTIATION"
```

```
# compute the deviation
```

```
res <- compute.kld(coembed = ce, genes.use = rownames(pbm3k),
  n.grids = 100, gene.set.list = gene.set.list[gs.names], gene.set.cutoff = 3,
  n.times = 100)
```

```
# To understand the output: The column 'kld'
# (log-transformed KL-divergence) reflects the differential
# distribution between the gene set and the background
# (genes.use) in the MCA space; the column 'rkld.mean'
# reflects the averaged differential distribution between a
# random gene set (size matched) and the background, and
# the 'rkld.sd' is the standard deviation (we do select
# random gene sets n.times). p-value for each 'kld' is
# calculated based on the distribution described by
# 'rkld.mean' and 'rkld.sd'. Multi-testing adjustment for
# p-values are performed (fdr).
```

```
head(res)
```

```
##                                gene.set                                kld
## 1          GOBP_B_1_B_CELL_DIFFERENTIATION -2.76100909185387
## 2                                GOBP_B_CELL_ACTIVATION -4.55912159221514
## 3 GOBP_B_CELL_ACTIVATION_INVOLVED_IN_IMMUNE_RESPONSE -4.29562204620625
## 4                                GOBP_B_CELL_APOPTOTIC_PROCESS -4.69208726585089
## 5                                GOBP_B_CELL_CHEMOTAXIS -2.05769059670985
## 6                                GOBP_B_CELL_DIFFERENTIATION -4.65412588468134
##  gene.set.length      rkld.mean      rkld.sd      p.value
## 1           5 -2.74132819857554 0.848185830448407 0.509256032900867
## 2          209 -6.29220423006622 0.406149331830666 9.90121182152379e-06
## 3           70 -5.19339367335727 0.443697911462374 0.0215167430284716
## 4           24 -4.23576911699505 0.494380829525733 0.821998844589698
## 5            5 -2.83916183983057 0.763260504992077 0.152950906335256
## 6          103 -5.61343842477255 0.358369288513474 0.00371553256471398
##      p.adj
## 1 6.146194e-01
## 2 6.930848e-05
## 3 6.275717e-02
## 4 8.394682e-01
## 5 2.974045e-01
## 6 1.444929e-02
```

```
# We can find out the deviated gene sets at alpha level
# equal to 0.05
gene.set.deviated <- res[res$p.adj < 0.05, ]$gene.set
gene.set.deviated
```

```
## [1] "GOBP_B_CELL_ACTIVATION"
## [2] "GOBP_B_CELL_DIFFERENTIATION"
## [3] "GOBP_B_CELL_MEDIATED_IMMUNITY"
## [4] "GOBP_B_CELL_PROLIFERATION"
## [5] "GOBP_B_CELL_RECEPTOR_SIGNALING_PATHWAY"
## [6] "GOBP_NEGATIVE_REGULATION_OF_B_CELL_PROLIFERATION"
## [7] "GOBP_POSITIVE_REGULATION_OF_B_CELL_ACTIVATION"
## [8] "GOBP_POSITIVE_REGULATION_OF_B_CELL_PROLIFERATION"
## [9] "GOBP_REGULATION_OF_B_CELL_ACTIVATION"
## [10] "GOBP_REGULATION_OF_B_CELL_PROLIFERATION"
## [11] "GOBP_REGULATION_OF_B_CELL_RECEPTOR_SIGNALING_PATHWAY"
```

```
# At this stage, we know that for all the B cell related GO
# terms, these ones demonstrated some patterns/specificity
# in our dataset. We do not need to know anything about the
# dataset itself (e.g., clustering information). Now, we
# can start from whichever term to fetch cells the most
# relevant to it.
```

2. Fetch cells for gene set of interest

```
# First, compute a nearest neighbor graph (edge list) in
# the MCA space
```

```

cells <- colnames(pbmc3k)
el <- compute.nn.edges(coembed = ce, nn.use = 300)

# Compute the relevance between each cell and the gene set
# (B_CELL_ACTIVATION)

cv <- run.rwr(el = el, gene_set = gene.set.list[["GOBP_B_CELL_ACTIVATION"]],
             cells = cells)
head(cv) # this gives a named numeric vector with the normalized label propagation probability, reflec

```

Knowing that some gene sets are enriched is good, but not enough. We are interested in which cells are the most relevant to the gene sets.

```

## AAACATACAACCAC AAACATTGAGCTAC AAACATTGATCAGC AAACCGTGCTTCCG AAACCGTGTATGCG
## 0.0001477042 0.0019110145 0.0002285931 0.0006810362 0.0001300983
## AAACGCACTGGTAC
## 0.0004111460

```

```

# We can also do this for multiple gene sets Again, using
# multi-threading is strongly recommended (mentioned above)
cv.df <- run.rwr.list(el = el, gene_set_list = gene.set.list[gene.set.deviated],
                    cells = cells)
cv.df[1:3, 1:3]

```

```

## GOBP_B_CELL_ACTIVATION GOBP_B_CELL_DIFFERENTIATION
## AAACATACAACCAC 0.0001477042 0.0002597976
## AAACATTGAGCTAC 0.0019110145 0.0019845892
## AAACATTGATCAGC 0.0002285931 0.0000000000
## GOBP_B_CELL_MEDIATED_IMMUNITY
## AAACATACAACCAC 2.898239e-05
## AAACATTGAGCTAC 1.536365e-03
## AAACATTGATCAGC 3.092756e-04

```

```

# Now we have the relevance between each cell and the gene
# set 'Fetching the cells for a gene set' is to binarize
# this relevance; we do this by fitting the label
# propagation probability to a bimodal distribution and
# find the antimode, and then use the antimode to binarize
# the data

```

```

cl <- compute.cell.label(cv)
head(cl)

```

```

## AAACATACAACCAC AAACATTGAGCTAC AAACATTGATCAGC AAACCGTGCTTCCG AAACCGTGTATGCG
## "negative" "positive" "negative" "negative" "negative"
## AAACGCACTGGTAC
## "negative"

```

```

table(cl)

```

```

## cl
## negative positive
## 2389 311

```

```
# We can also do this for multiple gene sets Again, using
# multi-threading is strongly recommended (mentioned above)
```

```
cl.df <- compute.cell.label.df(cv.df)
cl.df[1:3, 1:3]
```

```
##                GOBP_B_CELL_ACTIVATION GOBP_B_CELL_DIFFERENTIATION
## AAACATACAACCAC "negative"          "negative"
## AAACATTGAGCTAC "positive"         "positive"
## AAACATTGATCAGC "negative"         "negative"
##                GOBP_B_CELL_MEDIATED_IMMUNITY
## AAACATACAACCAC "negative"
## AAACATTGAGCTAC "positive"
## AAACATTGATCAGC "negative"
```

```
# with the binarized data, we can have an optional
# filtering step to further limit our gene terms with
# certain numbers of 'positive cells'. Sometimes a gene set
# only have a very small number of positive cells and we
# should be cautious on that. This step is included in the
# vignette 'spatial_example_10x_visium'.
```

```
# We can now visualize the data Compute the UMAP
# coordinates for visualization
```

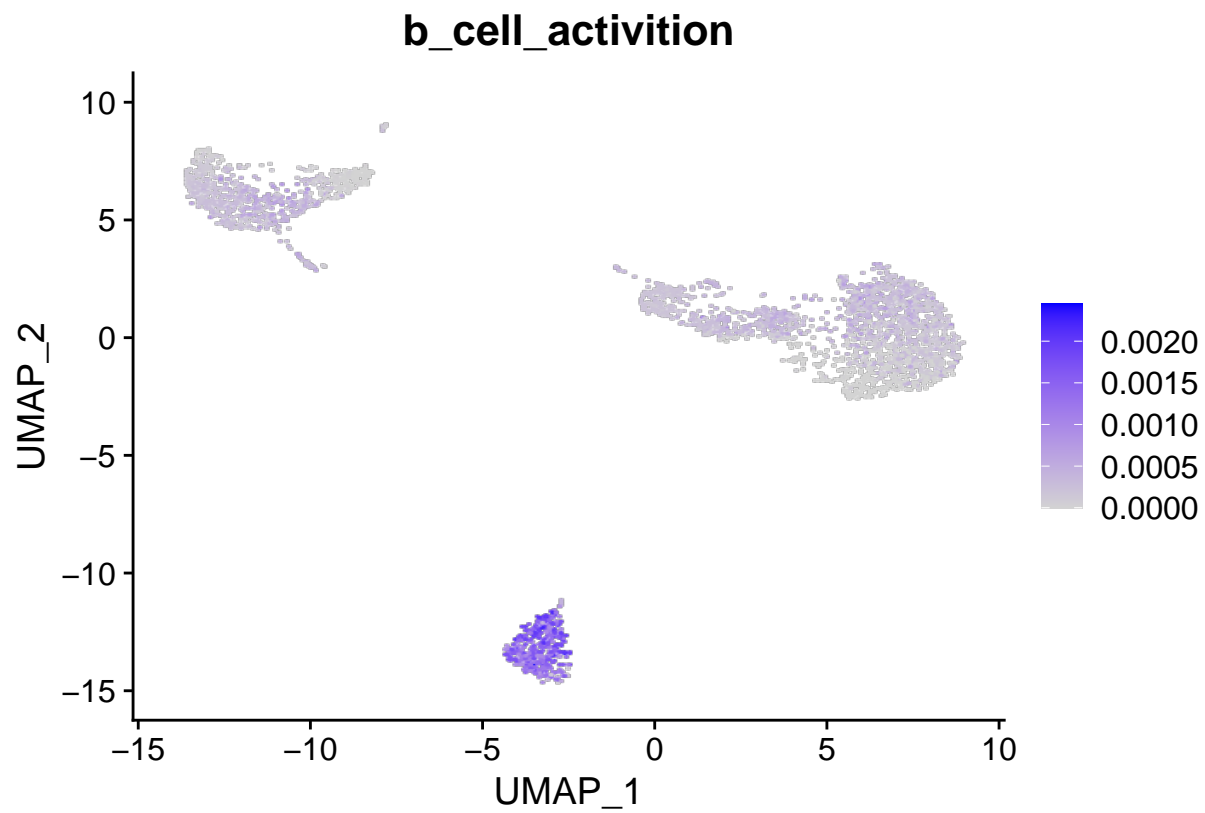
```
pbmc3k <- pbmc3k %>%
  FindVariableFeatures() %>%
  ScaleData() %>%
  RunPCA() %>%
  FindNeighbors() %>%
  RunUMAP(dims = 1:20)
```

```
# add the label propagation probability and binarized label
# to meta data
```

```
pbmc3k@meta.data$b_cell_activation <- cv[colnames(pbmc3k)]
pbmc3k@meta.data$b_cell_activation_bin <- cl[colnames(pbmc3k)]
```

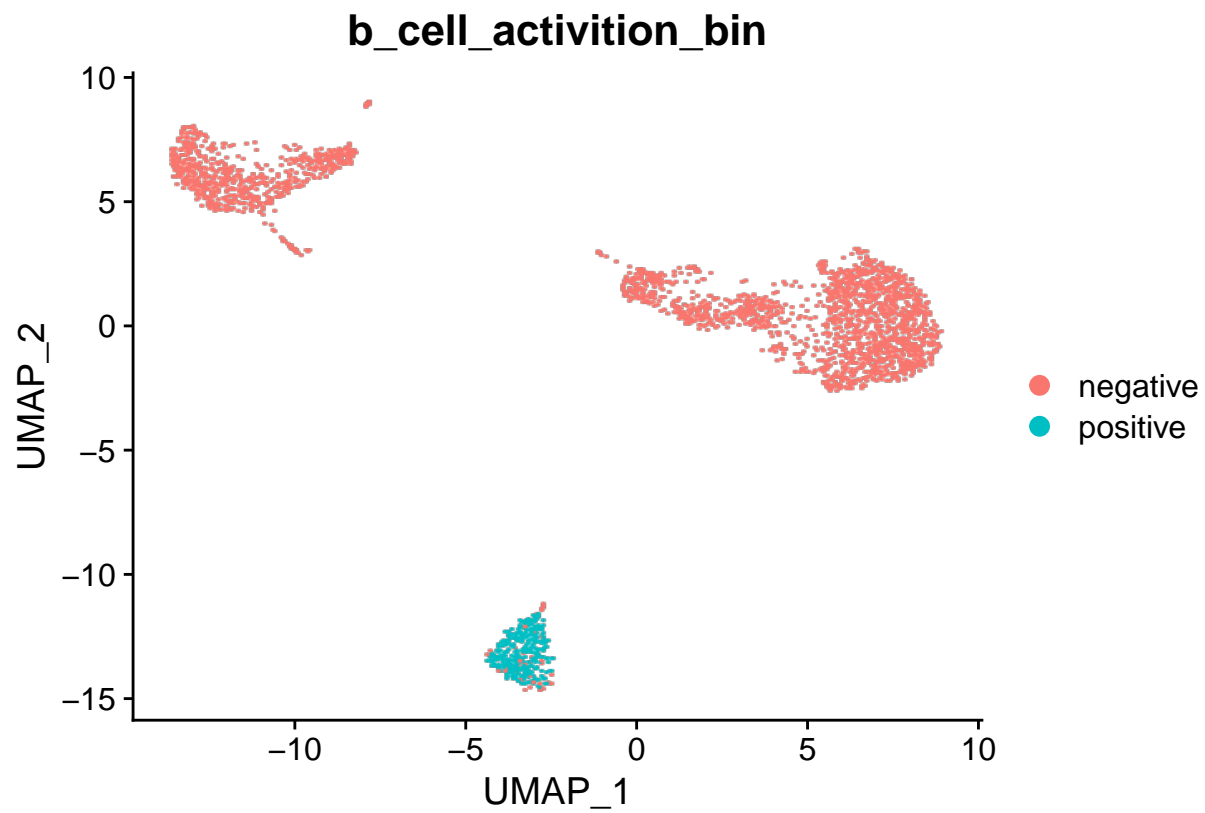
```
# Plot the label propagation probability
```

```
p1 <- FeaturePlot(pbmc3k, features = "b_cell_activation", raster = T)
p1
```

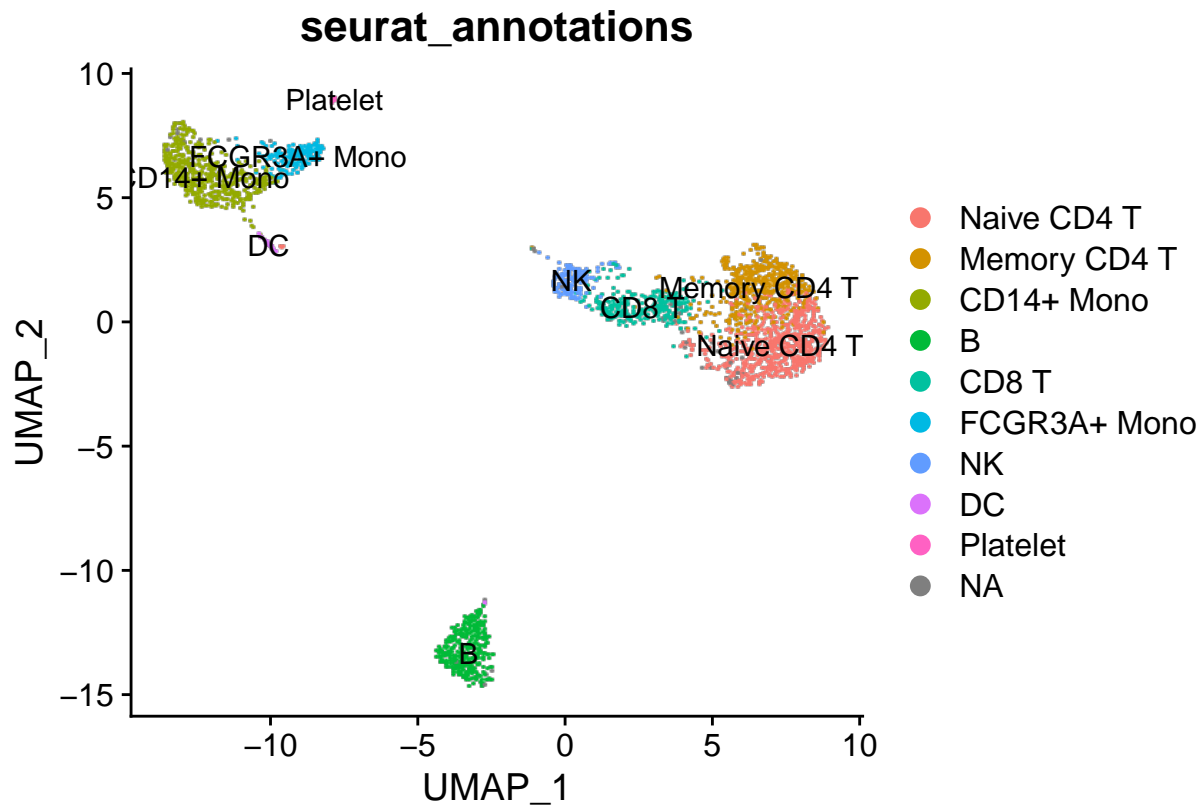



```
# Plot the binarized label
```

```
p2 <- DimPlot(pbmc3k, group.by = "b_cell_activation_bin", raster = T)  
p2
```



```
p3 <- DimPlot(pbmc3k, group.by = "seurat_annotations", label = T,  
  raster = T)  
p3
```



```
# This pbmc3k data is pre-annotated and the
# 'seurat_annotations' comes with the SeuratData We can see
# that the most relevant cells of 'B cell activation' are B
# cells, which is expected.
```

3. A brief example to compare gsdensity with GSEA

```
# We just use the 'B Cell activation' term use the escape
# package to run gsea
library(escape)

# default settings
ES.seurat <- enrichIt(obj = pbmc3k, gene.sets = list(B_CELL_ACTIVATION = gene.set.list[["GOBP_B_CELL_AC"]
groups = 1000, cores = 2)

## [1] "Using sets of 1000 cells. Running 3 times."
## Setting parallel calculations through a SnowParam back-end
## with workers=2 and tasks=100.
## Estimating ssGSEA scores for 1 gene sets.
## Setting parallel calculations through a SnowParam back-end
## with workers=2 and tasks=100.
## Estimating ssGSEA scores for 1 gene sets.
## Setting parallel calculations through a SnowParam back-end
```

```
## with workers=2 and tasks=100.
## Estimating ssGSEA scores for 1 gene sets.
```

```
cvrg <- ES.seurat$B_CELL_ACTIVATION
```

```
# gsea score for each cell
```

```
names(cvrg) <- rownames(ES.seurat)
```

```
head(cvrg)
```

```
## AAACATACAACCAC AAACATTGAGCTAC AAACATTGATCAGC AAACCGTGCTTCCG AAACCGTGTATGCG
```

```
##      0.2023091      0.7970123      0.3891273      0.3000894      0.4424885
```

```
## AAACGCACTGGTAC
```

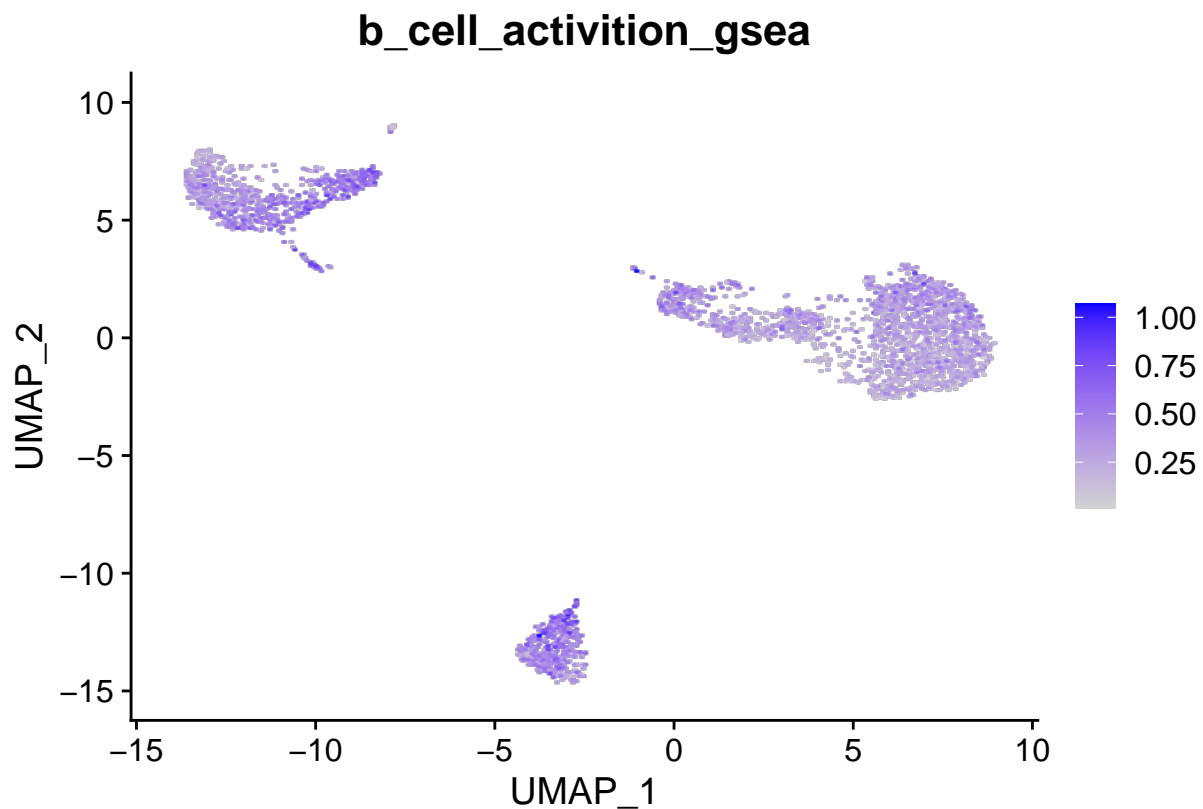
```
##      0.2116339
```

```
# add to meta.data
```

```
pbmc3k@meta.data$b_cell_activation_gsea <- cvrg[colnames(pbmc3k)]
```

```
# Plot
```

```
FeaturePlot(pbmc3k, c("b_cell_activation_gsea"), raster = T)
```



```
# we can see that the gsea score for this B cell related  
# term can not really distinguish B cells from others
```

```

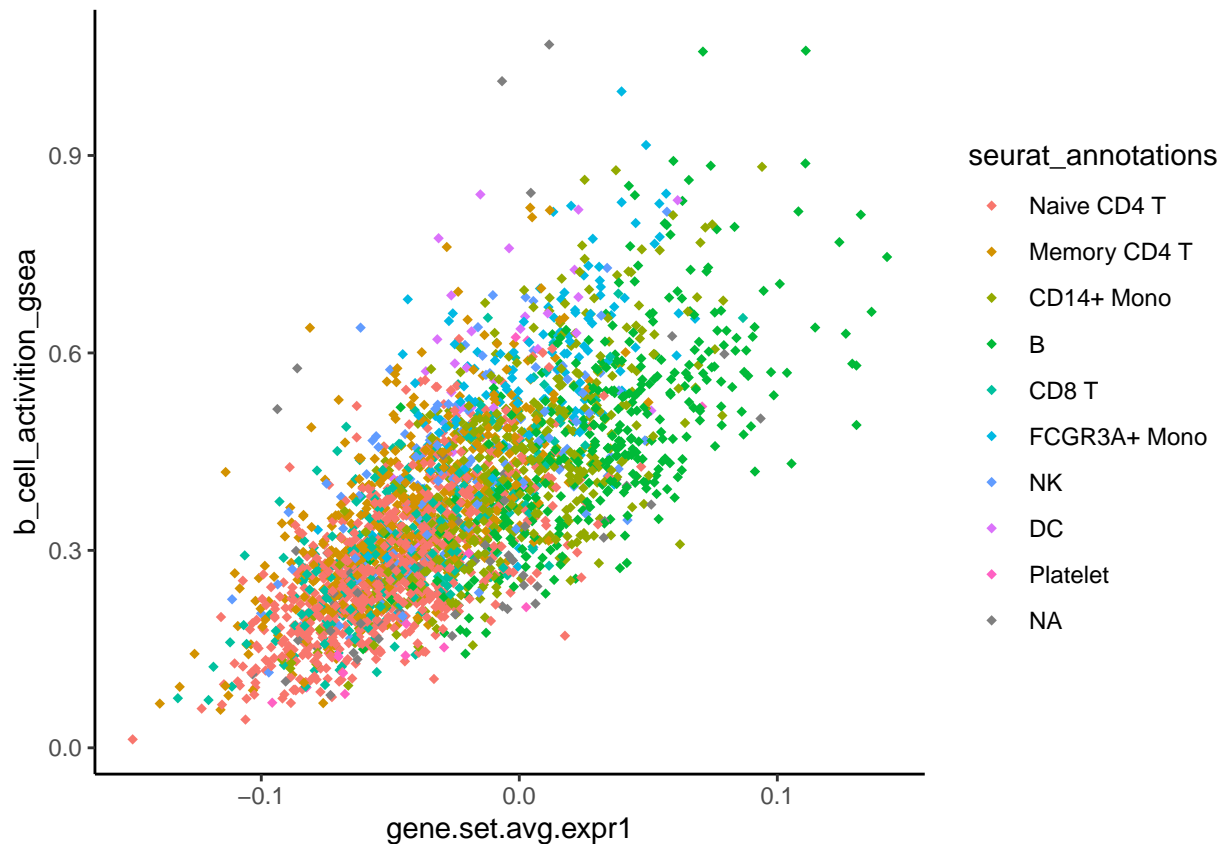
# We can look into the relationship between the gene set
# expression level and gsdensity/gsea use the
# 'AddModuleScore' function to evaluate the relative
# expression level of a gene set

# find common genes between the gene set and the scRNA-seq
# data
genes.of.interest <- intersect(rownames(pbmc3k), gene.set.list[["GOBP_B_CELL_ACTIVATION"]])
pbmc3k <- AddModuleScore(pbmc3k, features = list(genes.of.interest),
  ctrl = 20, name = "gene.set.avg.expr")
meta <- pbmc3k@meta.data

# Plot the relationship between the gene set expression and
# gsea score for each cell
p1 <- ggplot(meta, aes(x = gene.set.avg.expr1, y = b_cell_activation_gsea,
  color = seurat_annotatons)) + geom_point(shape = 18) + theme_classic()

p1

```

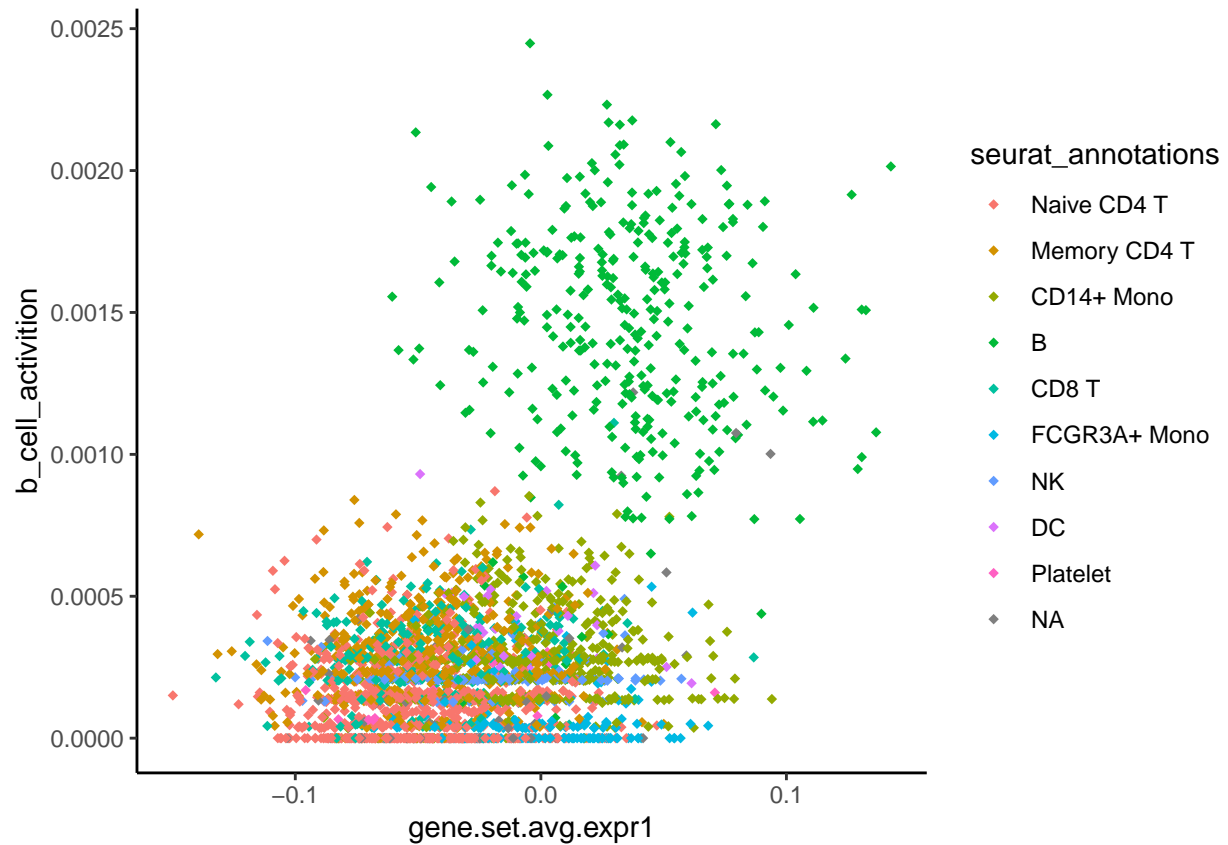


```

# Plot the relationship between the gene set expression and
# gsdensity score for each cell
p2 <- ggplot(meta, aes(x = gene.set.avg.expr1, y = b_cell_activation,
  color = seurat_annotatons)) + geom_point(shape = 18) + theme_classic()

p2

```



*# we can see that the gsea score is highly correlated to
 # the relative expression level of the gene set, but it is
 # not practical to find a way to distinguish real B cells
 # from others. With gsdensity, although some B cells do not
 # show higher gene set expression than others, they can be
 # distinguished using the gsdensity method*