Analysis of glioblastoma (GBM) scRNA-seq samples with scBubbletree

Simo Kitanovski (simo.kitanovski@uni-due.de)

Contents

Background 1

Data 1

Data processing 2

Step 1: determining the clustering resolution r for Louvain clustering 3

Step 2-3: clustering and hierarchical bubble grouping → bubbletree 5

Step 4: bubbletree visualization and annotation 5

Lets inspect the bubbletree . . . 5

Annotating the bubbletree with cell features . . . 6

What is the **cell type** composition of the bubbles? . . . 6

Compositional analysis: are all cells from different samples **evenly mixed** within bubbles? . . . 8

Quality control with *scBubbletree* . . . 10

Other cell features: **cell phase** and **gene expression** . . . 12

Summary 14

Session Info 14

Background

Here we will use *scBubbletree* to analyze a scRNA-seq data with over 45,000 cells from five human glioblastoma (GBM) patients[1]. From each patient we have paired samples from three tissues: tumor center, peripheral infiltration zone and blood. The aim of this vignette is to demonstrate that *scBubbletree* is capable of analyzing complex scRNA-seq data from solid tumors. In our analysis we compare our results with those presented by Schmassmann et al., 2023, and highlight the advantages of using *scBubbletree*.

Data

The publicly available scRNA-seq data[2] has been generated by 10X Genomics Chromium Single Cell Controller. The raw data (barcode-gene count matrix) is accompanied by supplementary meta-data. The meta data including the following cell-specific features:

- cell type labels (coarse- and fine-grained)
- cell phase labels

[1] Schmassmann et al., 2023 https://doi.org/10.7554/eLife.92678.2

[2] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE197543

- sample name
- tissue site
- quality metrics: % of transcripts from mitochondrial/ribosomal genes

Data processing

Data processing was performed with the R-package *Seurat*. Gene expressions were normalized (function `NormalizeData`), highly-variable genes were identified (function `FindVariableFeatures`), and the expressions were scaled and centered gene-wise (function `ScaleData`). Next, principal component analysis (PCA) was performed with the function `RunPCA` based on the 5,000 most variable genes.

In the data we saw that the first 15 principal components capture most of the variance in the data (data not shown), and the proportion of variance explained by each subsequent principal component was negligible. Thus, we used the single cell projections (embeddings) in 15-dimensional feature space, $A^{45,466\times 15}$.

```
# # Description: we used this script to download the raw data and process it
# # as explained in the above paragraph .This script can be used to generate
# # the data GBM data
#
# # create directory
# dir.create(path = "case_study/")
#
# # download the data from:
# https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE197543
#
# extract data:
# gzip -d GSE197543_UMIsMatrix.txt.gz
# gzip -d GSE197543_colData.txt.gz
#
# # read.csv
# u <- read.csv(file = "case_study/GSE197543_UMIsMatrix.txt", sep = "\t")
#
# # create Seurat object from the raw counts and append the meta data to it
# d <- Seurat::CreateSeuratObject(counts = u, project = 'GBM')
#
# # get meta
# meta <- read.csv(file = "data/CS_Glio/GSE197543_colData.txt", sep = "\t")
# meta_tissue <- do.call(rbind, strsplit(x = rownames(d@meta.data), split= "_"))
# d@meta.data <- cbind(d@meta.data, meta)
# d@meta.data$tissue <- meta_tissue[,4]
# d@meta.data$patient <- meta_tissue[,3]
# d@meta.data$cell_id <- 1:nrow(d@meta.data)
#
# # normalize data
# d <- NormalizeData(object = d, normalization.method = "LogNormalize")
#
# d <- FindVariableFeatures(object = d, selection.method="vst", nfeatures=5000)
#
# d <- ScaleData(object = d, features = VariableFeatures(object = d))
#
# d <- RunPCA(object = d, npcs = 50, features = VariableFeatures(object = d))
#
# # Save file to new folder
# save(d, file = "case_study/d.RData")
```

```
# # matrix A: the main input of scBubbletree
# GBM_A <- d@reductions$pca@cell.embeddings[, 1:15]
#
# save(A, file = "GBM_data/GBM_A.RData")
# save(d@meta.data, file = "GBM_data/GBM_meta.RData")
```

To run this vignette we need to load a few packages:

```
library(scBubbletree)
library(ggplot2)
library(ggtree)
library(patchwork)
```

But first, load the PCA matrix, meta-data and the matrix with normalized gene expressions:

```
# Load the data
# pca
load("GBM_data/GBM_A.RData")
# meta
load("GBM_data/GBM_meta.RData")
# gene expressions
load("GBM_data/GBM_exp.RData")
```

Step 1: determining the clustering resolution r for Louvain clustering

One important objective in scRNA-seq data analysis is to identify groups (clusters) of transcriptionally similar cells by clustering. *scBubbletree* offers a number of clustering methods (Louvain, Leiden, k-means, etc.), most of which are also implemented in *Seurat*. Here we will use the original Louvain method.

To do clustering we need two sets of inputs:

1. the PCA matrix
2. clustering resolution r

How to determine r?

scBubbletree allows us to determine r from the data based on the gap statistic, which is implemented in the function `get_r`. This function explained in greater detail in our manuscript. We will evaluate the gap statistic using the following vector of r values:

```
rs <- 10^seq(from = -3, to = 1, by = 0.1)
```

Next, we run the function `get_r`:

```
data_r <- get_r(rs = rs, x = GBM_A, algorithm = "original")
```

How does the gap curve behave as a function of r?

The gap increases rapidly between $k' = 5$ and $k' = 13$. (figure below, panel A). After $k' = 13$, the gap starts drops to values below 4. Next, the gap increases gradually to 4.13 at $k' = 25$, i.e. even though we nearly double the number of communities, the gap increases by only 0.13. Next, we see another dip in the gap, followed by another gradual increase to 4.22 at $k' = 45$.

From `get_r` we get a mapping between the tested resolutions `r` and the corresponding numbers of detected communities k' (figure below, panel B).

The resolution $r = 0.158$ ($k' = 13$) looks like a good first choice in order to identify the canonical cell types in the GBM data. If the user is interested in e.g. finding rare tumor cells, then we recommend using a higher

resolution: $r = 0.79$ ($k' = 25$) or even to $r = 2.51$ ($k' = 45$).

scBubbletree's clustering workflow is transparent. It demands from the user to think and evaluate different clustering resolutions and thereby to understand the data. The only cost is additional computational time required to run `get_r` or `get_k`.

```
((ggplot(data_r$gap_stats_summary)+
   geom_vline(xintercept = c(13, 25, 45), linetype = "dashed", col = "gray")+
   geom_line(aes(y = gap_mean, x = k), col = "darkgray")+
   geom_point(aes(y = gap_mean, x = k), size = 1, col = "black")+
   geom_errorbar(aes(y = gap_mean, x = k, ymin = L95, ymax = H95), width = 0.1)+
   xlab(label = "k'")+
   ylab(label = "Gap")+
   theme_bw(base_size = 10))|
  (ggplot(data_r$gap_stats_summary)+
     geom_vline(xintercept = c(13, 25, 45), linetype = "dashed", col = "gray")+
     geom_line(aes(x = k, y = r), col = "darkgray")+
     geom_point(aes(x = k, y = r), size = 1, col = "black")+
     ylab(label = "r'")+
     xlab(label = "k'")+
     scale_x_log10()+
     scale_y_log10()+
     theme_bw(base_size = 10)+
     annotation_logticks(base = 10, sides = "bl")))+
  plot_annotation(tag_levels = 'A')
```

The clustering resolution $k' = 13$ is comparable with the number of clusters identified by Schmassmann et al., 2023

Schmassmann et al., 2023, identified between $k = 10$ and $k = 22$ clusters of transcriptionally similar cells with hierarchical clustering:

> Clustering of cells was performed with hierarchical clustering on the Euclidean distances between cells (with Ward's criterion to minimize the total variance within each cluster Murtagh and Legendre, 2014; package cluster version 2.1.0). The number of clusters used for following analyses was identified by applying a dynamic tree cut (package dynamicTreeCut, version 1.63–1) (Langfelder et al., 2008), resulting in 10 with argument deepSplit set to 1, or 22 clusters with

argument deepSplit set to 2.

Step 2-3: clustering and hierarchical bubble grouping → bubbletree

Next, we perform Louvain clustering with $r = 0.158$ ($k' = 13$). Once the clusters (bubbles) are identified, we will arrange the bubbles in a bubbletree (hierarchical dendrogram) using hierarchical clustering with average linkage.

To construct the bubbletree, we will use $B = 300$ bootstrap iterations. In each iteration, we will draw from each bubble $N_{\text{eff}} = 100$ cells at random without replacement. The details of the bubbletree construction algorithm are provided in our manuscript.

Lets run `get_bubbletree_graph` with $r = 0.158$:

```
# graph-based clustering -> bubbletree
btd <- get_bubbletree_graph(x = GBM_A,
                            r = 0.158,
                            algorithm = "original",
                            hclust_method = "average",
                            hclust_distance = "euclidean",
                            cores = 5,
                            B = 300,
                            show_simple_count = TRUE,
                            N_eff = 100)
```

Step 4: bubbletree visualization and annotation

Lets inspect the bubbletree

We see two outstandingly large bubbles: 0 and 1. These bubbles account for about 50% of the cells in the GBM data. The smallest bubble 12 contains only 184 cells (0.4% of all cells).

Nearly all of the branches in the bubbletree are completely robust, i.e. they were observed consistently in B=300 bootstrap iterations. Only the branch between the bubble 3 and 2 is less robust and is found in less than half of the bootstrap dendrograms.

For biological interpretation of the bubbletree structure, we will have to annotate the bubbletree with cluster-specific summaries of different cell features (see next section).

```
btd$tree
```

Annotating the bubbletree with cell features

scBubbletree stacks bubble-specific summaries of categorical and numeric cell features with the bubbletree.

Here, we will use the function `get_cat_tiles` to generate heatmaps for each of the following categorical features, which are provided as part of the meta-data:

1. coarse-grained cell type labels
2. fine-grained cell type labels
3. cell phases
4. sample labels

We will use the input parameter `integrate_vertical = FALSE`. This means that we will show the composition of each bubble (y-axis) with respect to the categories (labels; x-axis) of each feature, i.e. the within-bubble relative frequencies of the different labels.

To visualize numeric features, such as mean gene expressions in each bubble, we will use the function `get_num_tiles`.

What is the cell type composition of the bubbles?

To answer this question, we will run the function `get_cat_tiles` using as input two categorical features (two columns in `GBM_meta`: CellAnnotation and CellAnnotationFine), which provide the predicted cell type labels of each cell at two levels of resolution: coarse-grained and fine-grained, respectively.

We also have the tissue site of each cell as a categorical feature (column tissue in data `GBM_meta`). We will also show the tissue-site composition within each bubble. We expect the some of the bubbles will be enriched with cells from the central tumor tissue samples, while others will be enriched with cells from the blood samples.

Lets run this function now:

```r
# cell type annotations
g_c <- get_cat_tiles(btd = btd,
                     f = GBM_meta$CellAnnotation,
                     round_digits = 1,
                     tile_text_size = 2,
                     integrate_vertical = FALSE)

# cell type 'fine' annotations
g_cf <- get_cat_tiles(btd = btd,
                      f = GBM_meta$CellAnnotationFine,
                      round_digits = 1,
                      tile_text_size = 2,
                      integrate_vertical = FALSE)

# tissue
g_t <- get_cat_tiles(btd = btd,
                     f = GBM_meta$tissue,
                     round_digits = 1,
                     tile_text_size = 2,
                     integrate_vertical = FALSE)
```

Next, we will join the bubbletree with these annotations into one visual:

```r
(btd$tree|g_c$plot|g_cf$plot|g_t$plot)+
    plot_layout(widths = c(1, 1, 1.5, 0.4))+
    plot_annotation(tag_levels = 'A')
```

Lets interpret this visual

First, >95% of the cells in bubbles 8, 5, 11, 10 and 12 are CD45- (non-hematopoietic cells). Second, the remaining bubbles are enriched with immune cells. This is consistent with the results in:

> "Using hierarchical clustering, the cells were partitioned into clusters (Figure 2-figure supplement 1A and B) which were then annotated into nine distinct cell types for the immune subset, including two transcriptionally distinct MG subsets (MG_1 and MG_2) and four cell types for the CD45-subset (Figure 2A; Figure 2-figure supplement 1C and D; Supplementary file 3)."

Importantly, the bubbletree conserves global transcriptional distances between these bubbles and splits the bubbles in two branches:

- lymphocytes: B-cells, T-cells, NK-cells, etc.
- non-lymphocytes: monocytes, neutrophils, microglia and macrophages

Third, we see two biologically interesting bubbles 3 and 6:

- bubble 3: tumor-infiltrating lymphocytes → the bubble has few PBMCs
- bubble 6: microglia and macrophages

Global distances between the clusters are **not** adequately conserved in the 2D t-SNE plot presented in Schmassmann et al., 2023 (below is the replicated version of Figure 2A from Schmassmann et al., 2023). Furthermore, the 2D t-SNE plot suffers from heavy overplotting. This challenge occurs despite the fact that the cells are split into three panels for each tissue site.

```
ggplot(data = GBM_meta)+
  facet_wrap(facets = ~tissue, nrow = 1)+
  geom_point(aes(x = tSNE1, y = tSNE2, col = CellAnnotation), size = 0.25)+
  theme_bw()+
  theme(legend.position = "top")+
  guides(colour = guide_legend(override.aes = list(size=2)))+
  scale_color_discrete(name = "tissue")
```

Compositional analysis: are all cells from different samples evenly mixed within bubbles?

When analyzing multi-sample scRNA-seq data, we have to ask ourselves the following questions:

- is bubble i composed of cells from all of the samples (even mixture)?
- is bubble i enriched with cells from only one sample j?

- what is the relative abundance of cells from different samples *within* a bubble?
- what is the relative abundance of cells from a specific sample *across* all bubbles?

Here, we have samples from **five patients**. From each patient we have paired samples from three tissues:

- tumor center
- peripheral infiltration zone
- blood

We assume that blood samples are similar across patients, i.e. they are made up from similar cell types. Hence, in sample i we expect similar relative abundance of cells across all bubbles.

This assumption may not be correct in terms of the samples from tumor center or peripheral infiltration zone. These samples are enriched with tumor cells, and we know that GBM are heterogeneous both within and between patients.

From Schmassmann et al., 2023 we get the following remarks about the degree of overlap of cells from the different patients across cell types (clusters):

> As we observed a good overlap of cells across patients for most of the dataset (see Methods; Figure 1-figure supplement 2B-F), we chose not to perform any correction for patient-specific effects
>
> A quantitative test of overlap of cells across patients was made using the CellMixS package (Lütge et al., 2021) (version 1.12.0) developed to quantify the effectiveness of batch correction methods. We used the cell-specific mixing (CMS) score, which highlighted a very good overlap across cells from different patients in the lymphoid compartment and for the monocytes. The myeloid compartment displayed a slightly elevated patient- specific structure, while it was most pronounced for the CD45 negative subset.

We replicated this analysis with *scBubbletree* (figure below). Our analysis with `get_cat_tiles` visualized the relative abundance of cells from different samples (samples IDs are available for each cell) across different bubbles by setting `integrate_vertical = TRUE` (panel B). We also show the within-bubble composition by setting `integrate_vertical = FALSE` (panel C).

```
g_p_v <- get_cat_tiles(btd = btd,
                       f = paste0(GBM_meta$tissue, '-', GBM_meta$patient),
                       round_digits = 1,
                       tile_text_size = 2,
                       integrate_vertical = TRUE)

g_p_h <- get_cat_tiles(btd = btd,
                       f = paste0(GBM_meta$tissue, '-', GBM_meta$patient),
                       round_digits = 1,
                       tile_text_size = 2,
                       integrate_vertical = FALSE)
```

```
(btd$tree|g_p_v$plot|g_p_h$plot)+
    plot_layout(widths = c(1, 1.3, 1.3))+
    plot_annotation(tag_levels = 'A')
```

New observation: skewed within-bubble distribution of sample IDs

First, we evaluated the samples derived from blood (middle part of panel B). Across-bubbles, the samples IDs have similar relative abundances for these samples. There were also exceptions:

- sample PBMC-616 is enriched with monocytes (cells from bubble 0)
- sample PBMC-617 is enriched with neutrophils (cells from bubble 9)
- From panel C we observe that about 88% of the bubble 9 cells belong to sample PBMC-617, i.e. the bulk of the cells from this bubble come from sample PBMC-617

Second, we evaluated the samples from tumor center and peripheral infiltration zone. In these samples we see skewed relative abundance of cells across the bubbles:

- in panel B we see that about 70% of the cells from sample center-604 and 10.6% of the cells from sample periphery-604 are assigned to bubble 5 (NPC-like tumor cells). According to panel C, nearly all of the cells from this bubble come from patient 604. The same can be said about bubbles 11, 10 and 12, i.e. few cells in these bubbles come from patients other than 604.
- according to panel C, bubble 8 is enrichmed with cells from the samples center-616 and center-617, and according to panel B, few cells from the remaining samples are assigned to this bubble.

In conclusion, scBubbletree visualizes the relative abundance of sample labels within and across bubbles as categorical features. The resulting visuals (panels B and C above) are rich with quantitative summaries, which facilitates rapid detection of compositional biases between the samples.

The bubbles with immune cells contained a mixture of cells from each patient. The bubbles with tumor cells were often patient-specific, i.e. they were made up from cells from individual patients. This is a realistic outcome given the high degree of GBM heterogeneity both within and between individuals.

Quality control with *scBubbletree*

Finally, nearly every scRNA-seq data analysis begins with quality control (QC). scBubbletree allows you to explore QC metrics commonly used by the community:

1. the number of unique genes detected in each cell

2. total number of molecules detected within a cell
3. the percentage of reads that map to the mitochondrial genome

Furthermore, any cell feature (numeric or categorical) that can be used for QC. These can be attached to the bubbletree to identify bubbles enriched with low-quality cells.

```
g_qc <- get_num_violins(btd = btd,
                        f = as.matrix(GBM_meta[, c("PercentMitochondrial",
                                                   "PercentRibosomal",
                                                   "NumberDetectedGenes")]))
```

```
(btd$tree|g_qc$plot)+
    plot_layout(widths = c(1, 2.2))+
    plot_annotation(tag_levels = 'A')
```

According to Schmassmann et al., 2023, low quality cells have been filtered from GBM data before the data was archived. Hence, we did not expect to see bubbles that are enriched with low-quality cells.

Nevertheless, we want to remind the reader that scBubbletree can be used for QC. In fact, we argue that the above visual is simpler and more informative than the one used by Schmassmann et al., 2023 which suffers from overplotting (presented as Figure 1-figure supplement 1 C-D; the quality feature PercentRibosomal is not visualized in the publication). We have replicated their visual below:

```
(ggplot(data = GBM_meta)+
    facet_wrap(facets = ~tissue, nrow = 1)+
    geom_point(aes(x = tSNE1, y = tSNE2, col = PercentMitochondrial), size=0.5)+
    theme_bw()+
    theme(legend.position = "top")+
    guides(colour = guide_legend(override.aes = list(size=2))))/
(ggplot(data = GBM_meta)+
    facet_wrap(facets = ~tissue, nrow = 1)+
    geom_point(aes(x = tSNE1, y = tSNE2, col = NumberDetectedGenes), size=0.5)+
    theme_bw()+
```

```
    theme(legend.position = "top")+
    guides(colour = guide_legend(override.aes = list(size=2))))
```

Other cell features: cell phase and gene expression

There is no limit to the number of type of cell features that can be used to annotate the bubbletree. In the next, we show how to 'attach' summaries of predicted cell phase labels (panel B) and gene expressions (panel C) to the bubbles.

Lets create these plots:

```
# cell cycle phase
g_p <- get_cat_tiles(btd = btd,
                     f = GBM_meta$CellCyclePhase,
                     round_digits = 1,
                     tile_text_size = 2,
                     integrate_vertical = FALSE)

# gene expression
```

```
g_e <- get_num_tiles(btd = btd,
                     fs = GBM_exp,
                     summary_function = "mean",
                     round_digits = 1,
                     tile_text_size = 2)
```

```
(btd$tree|g_p$plot|g_e$plot)+
    plot_layout(widths = c(1, 0.4, 2.55))+
    plot_annotation(tag_levels = 'A')
```

What is the dominant cell phase in the bubbles?

According to panel B, the dominant cell phases vary from bubble to bubble. The cells in the bubbles containing immune cells are predominantly in G1 phase, i.e. >73% of the cells in each of the bubbles 6, 4, 0, 9, 7, 1, 2 and 3 are in this phase. This makes sense from a biological point of view, i.e. most immune cells are likely resting, and in their quiescent state they do not need to proliferate, thus they remain in the G0/G1 phase until activated.

In contrast to this, tumor cells, are more likely to be found in S and G2/ M phases due to their constant division. This is exactly what we observe, i.e. we see higher fraction of S (synthesis phase) phase for cells in the bubbles containing tumor cell.

The quantitative summaries encoded in panel B makes the visual generated by scBubbletree more informative than the complementary t-SNE visual (see below), in which cells (dots) are color coded according to their cell phases (presented in Schmassmann et al., 2023 as Figure 1-figure supplement 1 E).

```
ggplot(data = GBM_meta)+
  facet_wrap(facets = ~tissue, nrow = 1)+
  geom_point(aes(x = tSNE1, y = tSNE2, col = CellCyclePhase), size = 0.5)+
  theme_bw()+
  theme(legend.position = "top")+
  guides(colour = guide_legend(override.aes = list(size=2)))+
```

```
scale_color_discrete(name = "tissue")
```

What is the average expression of marker genes in each bubble?

Next, we visualized the average expression values of 26 marker genes in each bubble (panel C). This is similar to figure 2B in Schmassmann et al., 2023. Numeric features, such as gene expressions, can be visualized with scBubbletree. For this we can use the functions `get_num_tiles` or `get_num_violins`. To generate panel C we used `get_num_tiles`.

We can evaluate panel C to double-check whether the cell type predictions are consistent with the gene expression of cell type characteristic markers.

Here are few examples of this analysis:

- CD14 is a marker of monocytes. CD14 has high average expression in cells from bubble 0 (≈ 1.2). We know from the cell type annotation visual shown earlier that bubble 0 is enriched with monocytes.
- NCAM1 is a marker of different tumors. NCAM1 has high average expression in cells from the tumor cell enriched bubbles 5 (≈ 1.7) and 8 (≈ 1.9).

Summary

scBubbletree promotes simple and transparent visual exploration of scRNA-seq. It is **not a black-box approach** and the user is encouraged to explore the data with different values of k and r or custom clustering solutions. Attaching of cell features to the bubbletree is necessary for biological interpretation of the individual bubbles and their relationships which are described by the bubbletree.

Session Info

```
sessionInfo()
FALSE R version 4.4.0 (2024-04-24)
FALSE Platform: x86_64-pc-linux-gnu
FALSE Running under: Ubuntu 22.04.4 LTS
```

```
FALSE
FALSE Matrix products: default
FALSE BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
FALSE LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3;  LAPACK version 3.10.0
FALSE
FALSE locale:
FALSE  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
FALSE  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
FALSE  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
FALSE  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
FALSE  [9] LC_ADDRESS=C               LC_TELEPHONE=C
FALSE [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
FALSE
FALSE time zone: Europe/Berlin
FALSE tzcode source: system (glibc)
FALSE
FALSE attached base packages:
FALSE [1] stats     graphics  grDevices utils     datasets  methods   base
FALSE
FALSE other attached packages:
FALSE [1] patchwork_1.2.0     ggtree_3.12.0       ggplot2_3.5.1
FALSE [4] scBubbletree_1.7.16 BiocStyle_2.32.0
FALSE
FALSE loaded via a namespace (and not attached):
FALSE   [1] deldir_2.0-4           pbapply_1.7-2          gridExtra_2.3
FALSE   [4] rlang_1.1.3            magrittr_2.0.3         RcppAnnoy_0.0.22
FALSE   [7] spatstat.geom_3.2-9    matrixStats_1.3.0      ggridges_0.5.6
FALSE  [10] compiler_4.4.0         png_0.1-8              vctrs_0.6.5
FALSE  [13] reshape2_1.4.4         stringr_1.5.1          pkgconfig_2.0.3
FALSE  [16] fastmap_1.2.0          labeling_0.4.3         utf8_1.2.4
FALSE  [19] promises_1.3.0         rmarkdown_2.27         purrr_1.0.2
FALSE  [22] xfun_0.44              cachem_1.1.0           aplot_0.2.2
FALSE  [25] jsonlite_1.8.8         goftest_1.2-3          later_1.3.2
FALSE  [28] spatstat.utils_3.0-4   BiocParallel_1.38.0    irlba_2.3.5.1
FALSE  [31] parallel_4.4.0         cluster_2.1.6          R6_2.5.1
FALSE  [34] ica_1.0-3              spatstat.data_3.0-4    stringi_1.8.4
FALSE  [37] RColorBrewer_1.1-3     reticulate_1.37.0      parallelly_1.37.1
FALSE  [40] scattermore_1.2        lmtest_0.9-40          Rcpp_1.0.12
FALSE  [43] knitr_1.47             tensor_1.5             future.apply_1.11.2
FALSE  [46] zoo_1.8-12             sctransform_0.4.1      httpuv_1.6.15
FALSE  [49] Matrix_1.7-0           splines_4.4.0          igraph_2.0.3
FALSE  [52] tidyselect_1.2.1       abind_1.4-5            rstudioapi_0.16.0
FALSE  [55] yaml_2.3.8             spatstat.random_3.2-3  spatstat.explore_3.2-7
FALSE  [58] codetools_0.2-20       miniUI_0.1.1.1         listenv_0.9.1
FALSE  [61] lattice_0.22-6         tibble_3.2.1           plyr_1.8.9
FALSE  [64] withr_3.0.0            shiny_1.8.1.1          treeio_1.28.0
FALSE  [67] ROCR_1.0-11            evaluate_0.23          Rtsne_0.17
FALSE  [70] gridGraphics_0.5-1     future_1.33.2          fastDummies_1.7.3
FALSE  [73] survival_3.7-0         proxy_0.4-27           polyclip_1.10-6
FALSE  [76] fitdistrplus_1.1-11    pillar_1.9.0           BiocManager_1.30.23
FALSE  [79] Seurat_5.1.0           KernSmooth_2.23-24     plotly_4.10.4
FALSE  [82] ggfun_0.1.5            generics_0.1.3         RcppHNSW_0.6.0
FALSE  [85] sp_2.1-4               munsell_0.5.1          scales_1.3.0
```

```
FALSE  [88] tidytree_0.4.6         xtable_1.8-4           globals_0.16.3
FALSE  [91] glue_1.7.0             lazyeval_0.2.2         tools_4.4.0
FALSE  [94] data.table_1.15.4      RSpectra_0.16-1        RANN_2.6.1
FALSE  [97] fs_1.6.4               leiden_0.4.3.1         dotCall64_1.1-1
FALSE [100] cowplot_1.1.3          grid_4.4.0             tidyr_1.3.1
FALSE [103] ape_5.8                colorspace_2.1-0       nlme_3.1-165
FALSE [106] cli_3.6.2              spatstat.sparse_3.0-3  spam_2.10-0
FALSE [109] fansi_1.0.6            viridisLite_0.4.2      dplyr_1.1.4
FALSE [112] uwot_0.2.2             gtable_0.3.5           yulab.utils_0.1.4
FALSE [115] digest_0.6.35          progressr_0.14.0       ggrepel_0.9.5
FALSE [118] ggplotify_0.1.2        farver_2.1.2           htmlwidgets_1.6.4
FALSE [121] SeuratObject_5.0.2     memoise_2.0.1          htmltools_0.5.8.1
FALSE [124] lifecycle_1.0.4        httr_1.4.7             mime_0.12
FALSE [127] MASS_7.3-60.2
```