Lab 1: Getting Familiar with Bioconductor and Ordination

Omics Data Science

Himel Mallick

Contents

Introduction to the R/Bioconductor data classes

Welcome to Lab 1! In this lab, we will familiarize ourselves with the Bioconductor world and some visualizations, which are usually the first step in the analysis of omics data.

While I expect that all of you are proficient with R, many may not have used (or even heard of) Bioconductor.

As introduced in the class, Bioconductor is an open-source and open-development project for the analysis and comprehension of high-throughput genomic data.

Similar to CRAN, which hosts over 13,000 packages to date, Bioconductor is a collection of over 2,000 R packages, some developed by a set of core developers, some contributed by the larger community.

R/Bioconductor packages can be installed via the specialized BiocManager R package, available on CRAN.

Bioconductor package versions are coordinated via a six-month release system.

The current version of Bioconductor, 3.18, requires R 4.3.2. For this module, you will need to have version 3.18 of Bioconductor installed.

Note: As a developer, you can only host your packages at either CRAN or Bioconductor but not both. Alternatively, you can host packages on GitHub.

Check out the OSCA Chapter 2 for more details.

Installing Packages the Bioconductor Way

Pseudo code (works for CRAN, GitHub, and Bioconductor packages):

```r
install.packages("BiocManager") #from CRAN
packages <- c("packagename", "githubuser/repository", "biopackage")
BiocManager::install(packages)
BiocManager::valid()  #check validity of installed packages
```

The `SummarizedExperiment` class

```r
library(SummarizedExperiment)
library(GenomicRanges)
library(airway)
```

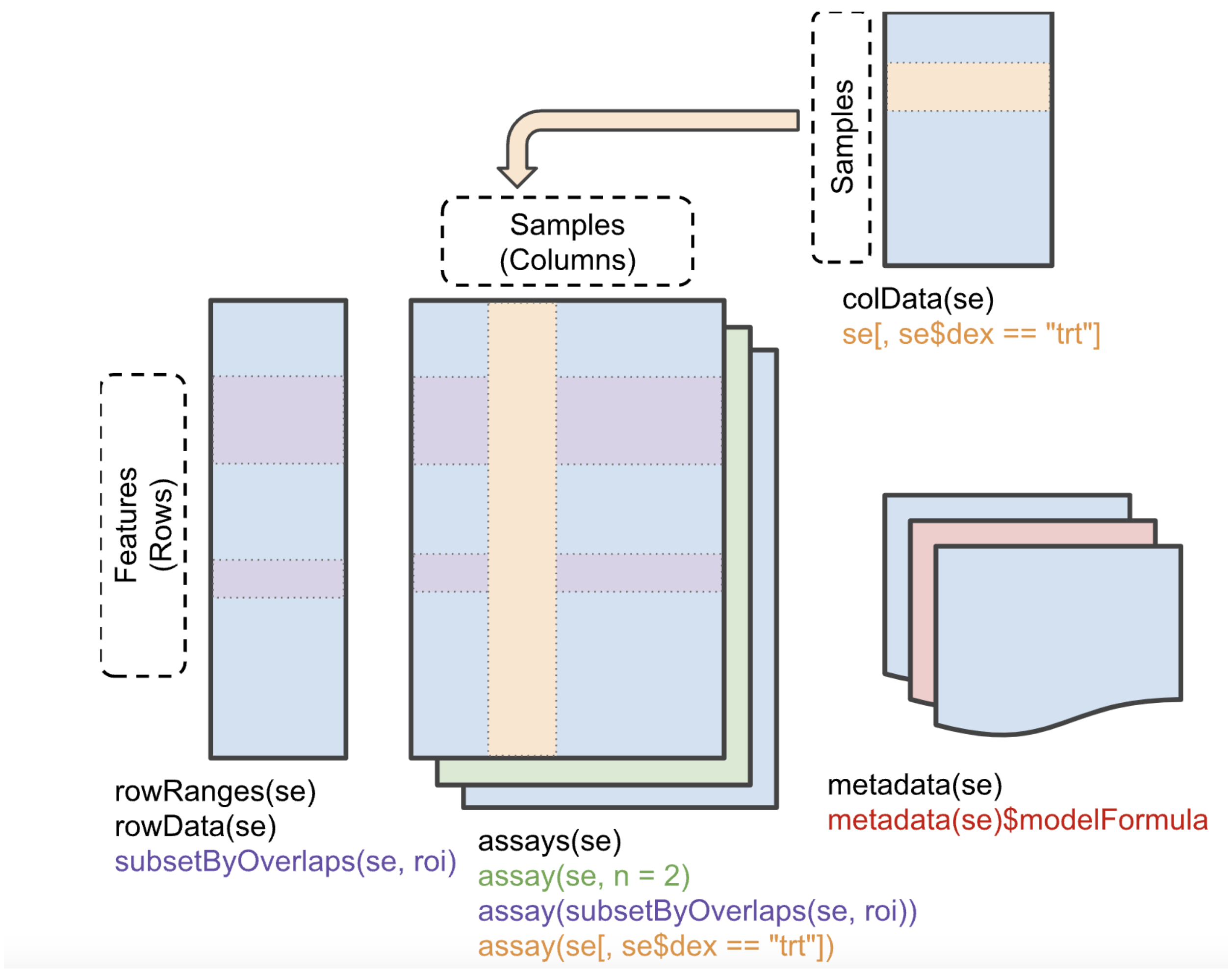

Figure 1: Summarized Experiment

One of the main strengths of the Bioconductor project lies in the use of a common data infrastructure that powers interoperability across packages.

Users should be able to analyze their data using functions from different Bioconductor packages without the need to convert between formats. To this end, the `SummarizedExperiment` class (from the *Summarized-*

Experiment package) serves as the common currency for data exchange across hundreds of Bioconductor packages.

This class implements a data structure that stores all aspects of the data - feature-by-sample omics measurements, per-sample metadata and per-feature annotation - and manipulate them in a synchronized manner.

Let's start with an example gene expression dataset.

```
data(airway)
airway
#> class: RangedSummarizedExperiment
#> dim: 63677 8
#> metadata(1): ''
#> assays(1): counts
#> rownames(63677): ENSG00000000003 ENSG00000000005 ... ENSG00000273492
#>   ENSG00000273493
#> rowData names(10): gene_id gene_name ... seq_coord_system symbol
#> colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
#> colData names(9): SampleName cell ... Sample BioSample
```

We can think of this class as a *container*, that contains several different pieces of data in so-called *slots* stitched together.

The *getter* methods are used to extract information from the slots and the *setter* methods are used to add information into the slots. These are the only ways to interact with the objects (rather than directly accessing the slots).

Depending on the object, slots can contain different types of data (e.g., numeric matrices, lists, etc.). We will here review the main slots of the SummarizedExperiment class as well as their getter/setter methods.

The `assays`

This is arguably the most fundamental part of the object that contains the omics measurements, and potentially other matrices with transformed data. We can access the *list* of matrices with the `assays` function and individual matrices with the `assay` function.

```
assay(airway)[1:3, 1:3]
#>                 SRR1039508 SRR1039509 SRR1039512
#> ENSG00000000003        679        448        873
#> ENSG00000000005          0          0          0
#> ENSG00000000419        467        515        621
```

You will notice that in this case, we have a regular matrix inside the object. More generally, any 'matrix-like' object can be used, including sparse matrices. We will see some of these other examples in later labs.

The `colData` and `rowData`

Conceptually, these are two data frames that annotate the columns and the rows of your assay, respectively.

One can interact with them as usual, e.g., by extracting columns or adding additional variables as columns.

```
colData(airway)
#> DataFrame with 8 rows and 9 columns
#>            SampleName     cell      dex    albut        Run avgLength
#>              <factor> <factor> <factor> <factor>   <factor> <integer>
#> SRR1039508 GSM1275862  N61311     untrt    untrt SRR1039508       126
#> SRR1039509 GSM1275863  N61311     trt      untrt SRR1039509       126
#> SRR1039512 GSM1275866  N052611    untrt    untrt SRR1039512       126
#> SRR1039513 GSM1275867  N052611    trt      untrt SRR1039513        87
#> SRR1039516 GSM1275870  N080611    untrt    untrt SRR1039516       120
#> SRR1039517 GSM1275871  N080611    trt      untrt SRR1039517       126
#> SRR1039520 GSM1275874  N061011    untrt    untrt SRR1039520       101
#> SRR1039521 GSM1275875  N061011    trt      untrt SRR1039521        98
#>            Experiment    Sample    BioSample
#>              <factor>  <factor>     <factor>
#> SRR1039508  SRX384345 SRS508568 SAMN02422669
#> SRR1039509  SRX384346 SRS508567 SAMN02422675
#> SRR1039512  SRX384349 SRS508571 SAMN02422678
#> SRR1039513  SRX384350 SRS508572 SAMN02422670
#> SRR1039516  SRX384353 SRS508575 SAMN02422682
#> SRR1039517  SRX384354 SRS508576 SAMN02422673
#> SRR1039520  SRX384357 SRS508579 SAMN02422683
#> SRR1039521  SRX384358 SRS508580 SAMN02422677
rowData(airway)
#> DataFrame with 63677 rows and 10 columns
#>                         gene_id     gene_name  entrezid   gene_biotype
#>                     <character>   <character> <integer>    <character>
#> ENSG00000000003 ENSG00000000003        TSPAN6        NA protein_coding
#> ENSG00000000005 ENSG00000000005          TNMD        NA protein_coding
#> ENSG00000000419 ENSG00000000419          DPM1        NA protein_coding
#> ENSG00000000457 ENSG00000000457         SCYL3        NA protein_coding
#> ENSG00000000460 ENSG00000000460      C1orf112        NA protein_coding
#> ...                         ...           ...       ...            ...
#> ENSG00000273489 ENSG00000273489 RP11-180C16.1        NA      antisense
#> ENSG00000273490 ENSG00000273490        TSEN34        NA protein_coding
#> ENSG00000273491 ENSG00000273491  RP11-138A9.2        NA        lincRNA
#> ENSG00000273492 ENSG00000273492    AP000230.1        NA        lincRNA
#> ENSG00000273493 ENSG00000273493  RP11-80H18.4        NA        lincRNA
#>                 gene_seq_start gene_seq_end              seq_name seq_strand
#>                      <integer>    <integer>           <character>  <integer>
#> ENSG00000000003       99883667     99894988                     X         -1
#> ENSG00000000005       99839799     99854882                     X          1
#> ENSG00000000419       49551404     49575092                    20         -1
#> ENSG00000000457      169818772    169863408                     1         -1
#> ENSG00000000460      169631245    169823221                     1          1
#> ...                        ...          ...                   ...        ...
#> ENSG00000273489      131178723    131182453                     7         -1
#> ENSG00000273490       54693789     54697585 HSCHR19LRC_LRC_J_CTG1          1
#> ENSG00000273491      130600118    130603315            HG1308_PATCH          1
#> ENSG00000273492       27543189     27589700                    21          1
#> ENSG00000273493       58315692     58315845                     3          1
#>                 seq_coord_system        symbol
#>                        <integer>   <character>
#> ENSG00000000003               NA        TSPAN6
#> ENSG00000000005               NA          TNMD
#> ENSG00000000419               NA          DPM1
#> ENSG00000000457               NA         SCYL3
#> ENSG00000000460               NA      C1orf112
#> ...                          ...           ...
#> ENSG00000273489               NA RP11-180C16.1
#> ENSG00000273490               NA        TSEN34
#> ENSG00000273491                NA  RP11-138A9.2
#> ENSG00000273492                NA    AP000230.1
#> ENSG00000273493                NA  RP11-80H18.4
```

Note the $ short cut.

```
identical(colData(airway)$cell, airway$cell)
#> [1] TRUE
airway$my_sum <- colSums(assay(airway))
colData(airway)
#> DataFrame with 8 rows and 10 columns
#>            SampleName     cell      dex    albut        Run avgLength
#>              <factor> <factor> <factor> <factor>   <factor> <integer>
#> SRR1039508 GSM1275862  N61311     untrt    untrt SRR1039508       126
#> SRR1039509 GSM1275863  N61311     trt      untrt SRR1039509       126
#> SRR1039512 GSM1275866  N052611    untrt    untrt SRR1039512       126
#> SRR1039513 GSM1275867  N052611    trt      untrt SRR1039513        87
#> SRR1039516 GSM1275870  N080611    untrt    untrt SRR1039516       120
#> SRR1039517 GSM1275871  N080611    trt      untrt SRR1039517       126
#> SRR1039520 GSM1275874  N061011    untrt    untrt SRR1039520       101
#> SRR1039521 GSM1275875  N061011    trt      untrt SRR1039521        98
#>            Experiment    Sample    BioSample    my_sum
#>              <factor>  <factor>     <factor> <numeric>
#> SRR1039508  SRX384345 SRS508568 SAMN02422669  20637971
#> SRR1039509  SRX384346 SRS508567 SAMN02422675  18809481
#> SRR1039512  SRX384349 SRS508571 SAMN02422678  25348649
#> SRR1039513  SRX384350 SRS508572 SAMN02422670  15163415
#> SRR1039516  SRX384353 SRS508575 SAMN02422682  24448408
#> SRR1039517  SRX384354 SRS508576 SAMN02422673  30818215
#> SRR1039520  SRX384357 SRS508579 SAMN02422683  19126151
#> SRR1039521  SRX384358 SRS508580 SAMN02422677  21164133
```

Ordination

PCA of Zeisel single-cell RNA-seq dataset

Let's first load the Zeisel single-cell RNA-seq dataset from the scRNAseq package. We will work with the normalized and transformed version of the data to generate the ordination plots.

```
suppressPackageStartupMessages({
  library(scRNAseq)
  library(BiocSingular)
  library(scran)
  library(scater)
})
sce.zeisel <- ZeiselBrainData()
sce.zeisel <- logNormCounts(sce.zeisel)
```

```
sce.zeisel <- fixedPCA(sce.zeisel, subset.row=NULL)
plotReducedDim(sce.zeisel, dimred="PCA", colour_by="level1class")
```

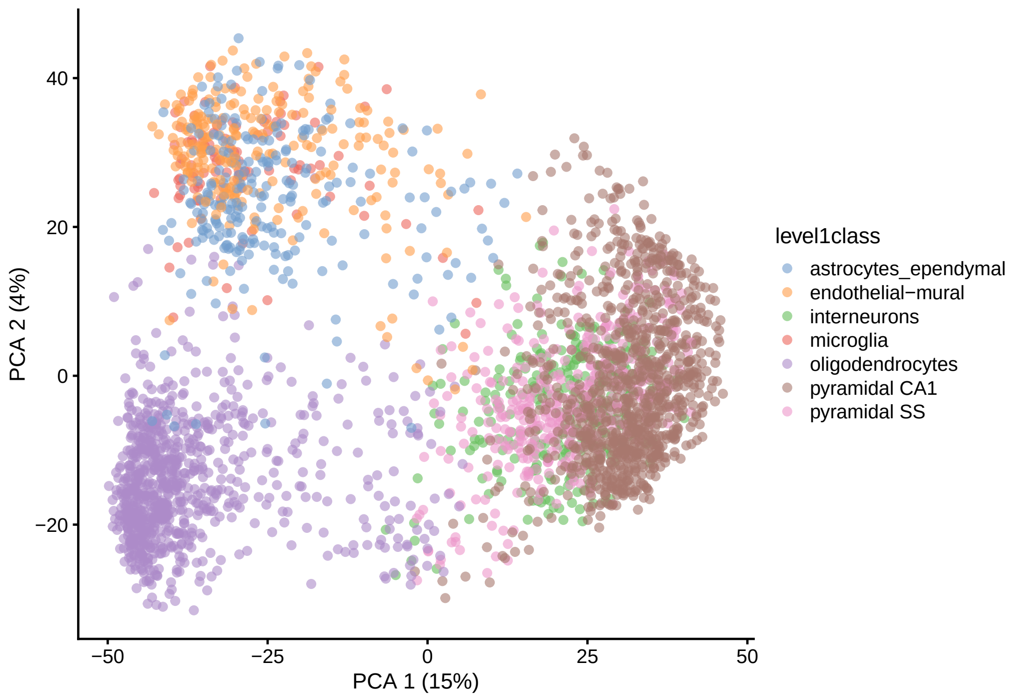

Figure 2: Principal Components Analysis of Zeisel dataset

t-SNE of the same dataset

```
sce.zeisel <- runTSNE(sce.zeisel, dimred="PCA")
plotReducedDim(sce.zeisel, dimred="TSNE", colour_by="level1class")
```

UMAP of the same dataset

```
sce.zeisel <- runUMAP(sce.zeisel, dimred="PCA")
plotReducedDim(sce.zeisel, dimred="UMAP", colour_by="level1class")
```

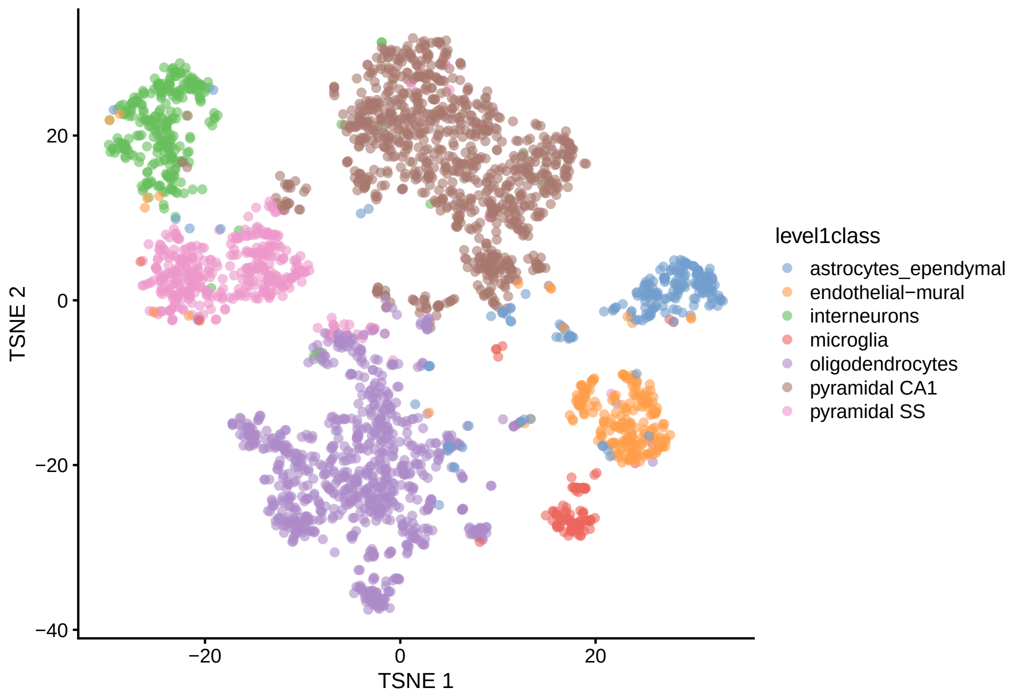

Figure 3: t-SNE clustering of Zeisel dataset

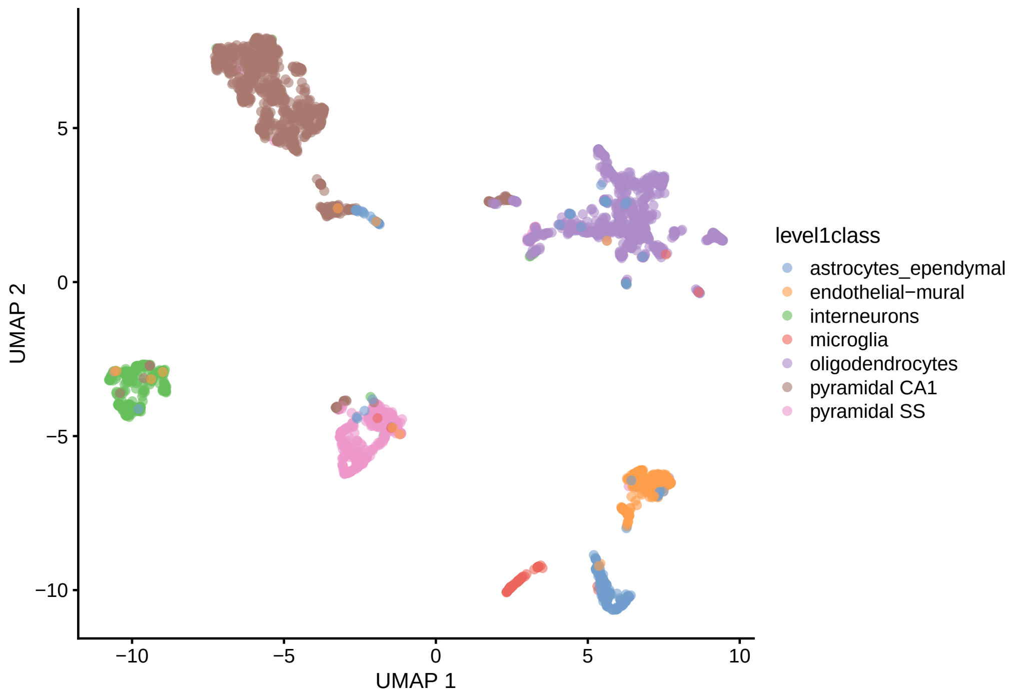

Figure 4: UMAP representation of the Zeisel dataset

Homework 1

- Install the `curatedMetagenomicData` package and explore the associated vignettes to extract the `HMP_2019_ibdmdb` dataset from the 2019 Nature paper.
- Among multiple assays (data types) available, consider the relative abundances to calculate the number of features and samples in this dataset. What is the disease of interest?
- Explore the OMA Chapter 7 to select a suitable distance (diversity) metric for HMP relative abundances. Then, generate UMAP, t-SNE, and PCA plots using the chosen metric. Color the plots according to the disease variable. Summarize your findings in one or two sentences.

Helpful Links

- The Bioconductor Project carpentries course
- OSCA Chapter 4 Dimensionality Reduction
- OMA Chapter 7 Community Similarity

Session Info

```
sessionInfo()
#> R version 4.4.2 (2024-10-31)
#> Platform: aarch64-apple-darwin20
#> Running under: macOS Sequoia 15.2
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> time zone: Asia/Kolkata
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] scater_1.32.1               ggplot2_3.5.1
#>  [3] scran_1.32.0                scuttle_1.14.0
#>  [5] BiocSingular_1.20.0         scRNAseq_2.18.0
#>  [7] SingleCellExperiment_1.26.0 airway_1.24.0
#>  [9] SummarizedExperiment_1.34.0 Biobase_2.64.0
#> [11] GenomicRanges_1.56.2        GenomeInfoDb_1.40.1
#> [13] IRanges_2.38.1              S4Vectors_0.42.1
#> [15] BiocGenerics_0.50.0         MatrixGenerics_1.16.0
#> [17] matrixStats_1.4.1
#>
#> loaded via a namespace (and not attached):
#>   [1] rstudioapi_0.17.1         jsonlite_1.8.9
#>   [3] magrittr_2.0.3            ggbeeswarm_0.7.2
#>   [5] GenomicFeatures_1.56.0    gypsum_1.0.1
#>   [7] farver_2.1.2              rmarkdown_2.29
#>   [9] BiocIO_1.14.0             zlibbioc_1.50.0
#>  [11] vctrs_0.6.5               memoise_2.0.1
#>  [13] Rsamtools_2.20.0          DelayedMatrixStats_1.26.0
#>  [15] RCurl_1.98-1.16           tinytex_0.54
#>  [17] htmltools_0.5.8.1         S4Arrays_1.4.1
#>  [19] AnnotationHub_3.12.0      curl_6.0.1
#>  [21] BiocNeighbors_1.22.0      Rhdf5lib_1.26.0
#>  [23] SparseArray_1.4.8         rhdf5_2.48.0
#>  [25] alabaster.base_1.4.2      alabaster.sce_1.4.0
#>  [27] httr2_1.0.7               cachem_1.1.0
#>  [29] GenomicAlignments_1.40.0  igraph_2.1.2
#>  [31] lifecycle_1.0.4           pkgconfig_2.0.3
#>  [33] rsvd_1.0.5                Matrix_1.7-1
#>  [35] R6_2.5.1                  fastmap_1.2.0
#>  [37] GenomeInfoDbData_1.2.12   digest_0.6.37
#>  [39] colorspace_2.1-1          AnnotationDbi_1.66.0
#>  [41] dqrng_0.4.1               irlba_2.3.5.1
#>  [43] ExperimentHub_2.12.0      RSQLite_2.3.9
#>  [45] beachmat_2.20.0           labeling_0.4.3
#>  [47] filelock_1.0.3            httr_1.4.7
#>  [49] abind_1.4-8               compiler_4.4.2
#>  [51] bit64_4.5.2               withr_3.0.2
#>  [53] BiocParallel_1.38.0       viridis_0.6.5
#>  [55] DBI_1.2.3                 HDF5Array_1.32.1
#>  [57] alabaster.ranges_1.4.2    alabaster.schemas_1.4.0
#>  [59] rappdirs_0.3.3            DelayedArray_0.30.1
#>  [61] rjson_0.2.23              bluster_1.14.0
#>  [63] tools_4.4.2               vipor_0.4.7
#>  [65] beeswarm_0.4.0            glue_1.8.0
#>  [67] restfulr_0.0.15           rhdf5filters_1.16.0
#>  [69] grid_4.4.2                Rtsne_0.17
#>  [71] cluster_2.1.8             generics_0.1.3
#>  [73] gtable_0.3.6              ensembldb_2.28.1
#>  [75] ScaledMatrix_1.12.0       metapod_1.12.0
#>  [77] XVector_0.44.0            ggrepel_0.9.6
#>  [79] BiocVersion_3.19.1        pillar_1.10.0
#>  [81] limma_3.60.6              dplyr_1.1.4
#>  [83] BiocFileCache_2.12.0      lattice_0.22-6
#>  [85] FNN_1.1.4.1               rtracklayer_1.64.0
#>  [87] bit_4.5.0.1               tidyselect_1.2.1
#>  [89] locfit_1.5-9.10           Biostrings_2.72.1
#>  [91] knitr_1.49                gridExtra_2.3
#>  [93] ProtGenerics_1.36.0       edgeR_4.2.2
#>  [95] xfun_0.49                 statmod_1.5.0
#>  [97] UCSC.utils_1.0.0          lazyeval_0.2.2
#>  [99] yaml_2.3.10               evaluate_1.0.1
#> [101] codetools_0.2-20          tibble_3.2.1
#> [103] alabaster.matrix_1.4.2    BiocManager_1.30.25
#> [105] cli_3.6.3                 uwot_0.2.2
#> [107] munsell_0.5.1             Rcpp_1.0.13-1
#> [109] dbplyr_2.5.0              png_0.1-8
#> [111] XML_3.99-0.17             parallel_4.4.2
#> [113] blob_1.2.4                AnnotationFilter_1.28.0
#> [115] sparseMatrixStats_1.16.0  bitops_1.0-9
#> [117] viridisLite_0.4.2         alabaster.se_1.4.1
#> [119] scales_1.3.0              crayon_1.5.3
#> [121] rlang_1.1.4               cowplot_1.1.3
#> [123] KEGGREST_1.44.1
```