PROTOCOL

Blood Science

OPEN

A pipeline for single-cell chromatin accessibility data analysis

Mengke Zhang[a,b], Changya Chen[a,b,*]

[a]*State Key Laboratory of Experimental Hematology, National Clinical Research Center for Blood Diseases, Haihe Laboratory of Cell Ecosystem, Institute of Hematology & Blood Diseases Hospital, Chinese Academy of Medical Sciences & Peking Union Medical College, Tianjin 300020, China;* [b]*Tianjin Institutes of Health Science, Tianjin 301600, China*

Abstract

Single-cell chromatin accessibility analysis enables high-resolution dissection of regulatory elements and gene regulatory mechanisms. However, standardized and comprehensive analysis workflows remain limited. In this study, we present a streamlined pipeline for analyzing single-cell chromatin accessibility data. The workflow begins with data preprocessing using single-cell assay for transposase-accessible chromatin (scATAC)-pro or Cell Ranger ATAC, followed by peak calling with MACS2 and differential accessibility analysis to detect open chromatin regions and perform differential accessibility analysis to highlight regulatory differences among cell populations. Transcription factor activity is then inferred using chromVAR, incorporating motif enrichment and footprinting analysis. Finally, SCENIC+ is applied to reconstruct transcriptional regulatory networks, enabling in-depth exploration of epigenetic mechanisms at the single-cell level. This integrative approach offers a robust framework for decoding the regulatory landscape and understanding cellular heterogeneity in complex biological systems.

Key Words: Single-cell ATAC sequencing

1. INTRODUCTION

Various approaches have been developed to study chromatin accessibility, such as deoxyribonuclease I hypersensitivity sequencing (DNase-seq),[1–3] formaldehyde-assisted isolation of regulatory elements (FAIRE-seq),[4] micrococcal nuclease digestion with sequencing (MNase-seq),[5,6] assay for transposase-accessible chromatin with high-throughput sequencing (ATAC-seq),[7] and nucleosome occupancy and methylome sequencing (NOMe-seq).[8] ATAC-seq is a method for mapping chromatin accessibility genome-wide using a hyperactive Tn5 transposase, which inserts sequencing adapters into accessible chromatin regions.[9] ATAC-seq is faster, simpler, and requires fewer cells than DNase-seq and MNase-seq, which are more time-consuming and costly due to their large cell requirements and separate assays.[7] However, bulk ATAC-seq is unable to determine chromatin accessibility at the single-cell level, thereby limiting its application in identifying cell subpopulations. To address these limitations, single-cell ATAC-seq (scATAC-seq) has been developed. Leveraging the advantages of the ATAC-seq protocol, scATAC-seq can measure chromatin accessibility on the single-cell level and the data richness of single-cell genomics has been substantially expanded.[10] scATAC-seq data has become a popular tool for examining chromatin accessibility at the single-cell level. scATAC-seq has emerged as a powerful approach for dissecting cell type-specific regulatory programs, identifying rare cell populations, and inferring gene regulatory networks (GRNs).

However, scATAC-seq data analysis poses unique computational challenges, necessitating robust pipelines for preprocessing, peak calling, differential accessibility analysis, motif enrichment, and regulatory network reconstruction.

Current pipelines for scATAC-seq analysis offer a diverse toolkit, each with unique strengths and limitations. ArchR stands out for its scalability and rich visualization features, making it ideal for large datasets, though it can be memory-intensive and primarily R-based.[11] Signac,[12] as part of the Seurat ecosystem, excels at multi-omic integration with scRNA-seq and offers flexibility, but may struggle with very large datasets and lacks some automation. SnapATAC[13] is highly efficient and fast, particularly well-suited for large-scale clustering, yet its bin-based approach can lead to lower resolution at the peak level. cisTopic[14] provides interpretable topic modeling for regulatory programs but may have limitations in scalability and integration with other data types. MAESTRO[15] is valuable for joint scRNA-seq and scATAC-seq analysis, but can be complex to configure. Meanwhile, Cell Ranger ATAC is user-friendly and standardized for 10x data, but lacks flexibility for non-10x platforms or advanced customization. scATAC-pro[16] is a comprehensive tool for processing, analyzing, and visualizing single-cell chromatin accessibility sequencing data. Overall, tool choice depends on specific goals such as integration, scale, or user expertise, and no single pipeline yet dominates across all criteria. The current protocol provides a pipeline for downstream bioinformatics analysis of scATAC-seq (**Fig. 1**).

*Address correspondence: Changya Chen, State Key Laboratory of Experimental Hematology, National Clinical Research Center for Blood Diseases, Haihe Laboratory of Cell Ecosystem, Institute of Hematology & Blood Diseases Hospital, Chinese Academy of Medical Sciences & Peking Union Medical College, Tianjin 300020, China. E-mail address: chenchangya@ihcams.ac.cn (C. Chen).

Conflict of interest: The authors declare that they have no conflict of interest.

This work was supported by the Chinese Academy of Medical Sciences (CAMS) Innovation Funds for Medical Sciences (2024-I2M-3-001).

Blood Science (2026) 8, 1–16:e00259.

Received April 25, 2025; Accepted August 4, 2025.

http://dx.doi.org/10.1097/BS9.0000000000000259

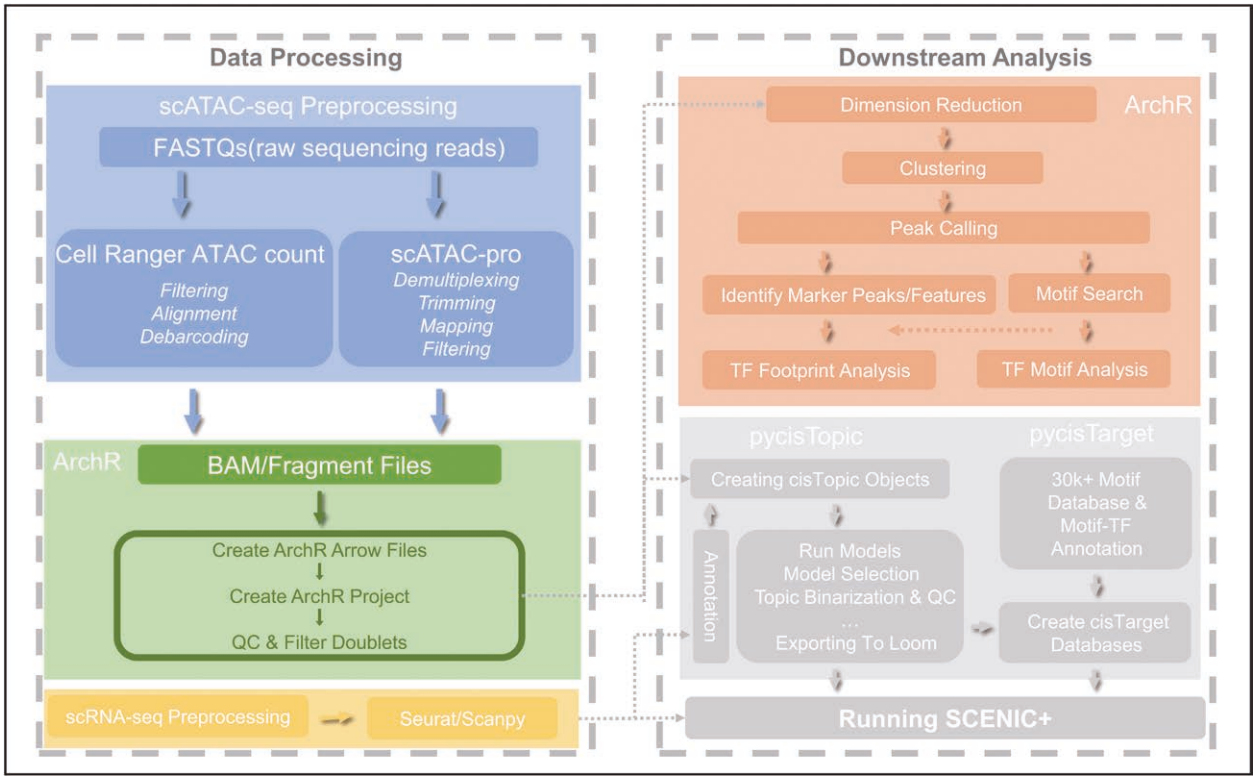

Figure 1. Schematic workflow for single-cell ATAC-seq data processing and downstream analysis. The analysis pipeline begins with raw sequencing data (FASTQs) processed using Cell Ranger ATAC and scATAC-pro for alignment, demultiplexing, trimming, mapping, and filtering. ArchR is then used to create Arrow files and projects, followed by quality control and doublet filtering. Dimensionality reduction and clustering enable identification of cell populations. Downstream analyses include peak calling, identification of marker peaks and features, TF motif search, and TF footprint analysis. ATAC = assay for transposase-accessible chromatin, TF = transcription factor.

2. PREPARATION

For data analysis, it is essential to have access to a computational server or a high-performance computing environment, since handling data at this scale can be too demanding for most regular computers. A fundamental understanding of command-line tools and bioinformatics is required for implementing the pipeline.

Hardware requirements:

- High-performance computing server (minimum 64 GB RAM recommended).
- Linux-based system (Ubuntu 20.04 or CentOS preferred).

Data requirements:

- Single-cell ATAC-seq raw sequencing reads in FASTQ format: 5k peripheral blood mononuclear cells (PBMCs) from a Healthy Donor (Next GEM v1.1) (5k PBMCs from a Healthy Donor [Next GEM v1.1]—10x Genomics).
- Cryopreserved human PBMCs from a healthy female donor aged 25 were obtained by 10x Genomics from AllCells. Epi Multiome ATAC + Gene Expression dataset analyzed using Cell Ranger ARC 2.0.0 (PBMC from a Healthy Donor—Granulocytes Removed Through Cell Sorting [3k]—10x Genomics).
- Genome reference (eg, human hg38, mouse mm10): Download the reference genome. Download the appropriate reference genome file from the 10x Genomics official website (https://support.10xgenomics.com/single-cell-atac/software/downloads/latest) based on the species studied. For example, download the GRCh38 reference genome for human data.

3. KEY RESOURCES TABLE

Software and algorithms

Cell Ranger ATAC(v2.1.0)	10x Genomics	https://support.10xgenomics.com/single-cell-atac/software/pipelines/latest/what-is-cell-ranger-atac
scATAC-pro(1.5.2)	Yu et al[16]	https://github.com/wbaopaul/scATAC-pro
R (v4.3.1)	R Core Team[17]	https://www.r-project.org
ArchR(v1.0.3)	Granja et al[11]	https://www.nature.com/articles/s41588-021-00790-6
MACS2(2.2.9.1)	Zhang et al[18]	https://github.com/macs3-project/MACS
ggplot2(3.5.1)	Wickham[19]	https://cran.r-project.org/web/packages/ggplot2/index.html
chromVAR(v1.24.0)	Schep et al[20]	https://github.com/GreenleafLab/chromVAR
SCENIC+(1.0a2)	Bravo Gonzalez-Blas et al[21]	https://scenicplus.readthedocs.io/en/latest/
Python(3.11.8)	Python Core Team[22]	https://www.python.org/

4. STEP-BY-STEP METHOD DETAILS

Step 1: Data preprocessing with Cell Ranger ATAC

Before analysis, ensure Cell Ranger ATAC is installed, the reference genome files are downloaded, and the sequencing-derived FASTQ files are prepared.

Install Cell Ranger ATAC Refer to the official documentation(https://support.10xgenomics.com/single-cell-atac/software/pipelines/latest/installation) for installation.

1.1 Create a working directory and extract data

```
>mkdir scATAC_analysis
>cd scATAC_analysis
```

1.2 Run Cell Ranger ATAC to generate single-nucleus accessibility counts

For the subsequent analysis in this tutorial, we will utilize the dataset titled *5k PBMCs from a Healthy Donor (Next GEM v1.1)* provided by 10x Genomics.

```
> cellranger-atac count
--id=atac_pbmc_5k_analysis \
--reference=/path/to/reference/refdata-
  cellranger-arc-GRCh38-2020-A-2.0.0/ \
--fastqs=/path/to/fastq_files/atac_pbmc_5k_
  nextgem_fastqs \
--sample= atac_pbmc_5k_nextgem \
--localcores=8 \
--localmem=64
```

1.3 Check Output Files

- outs/summary.csv: Quality control statistics
- outs/filtered_peak_bc_matrix.h5: Cell-by-peak matrix
- outs/cloupe.cloupe: Visualization file for Loupe Browser
- outs/fragments.tsv.gz: Input File Types in ArchR

Note: Cell Ranger ATAC is tightly integrated with the data format and platform of 10x Genomics single-cell ATAC-seq, making it potentially incompatible with data from other sources. Therefore, we introduce an alternative tool here: scATAC-pro.

Step 2: Data preprocessing with scATAC-Pro

scATAC-pro is an integrated open-source computational toolkit designed for processing, analyzing, and visualizing sequencing datasets from single-cell chromatin accessibility experiments.[16] The scATAC-pro framework comprises 2 core functional units: a data processing unit and a downstream analysis unit. The data processing unit accepts raw FASTQ files as input and generates critical outputs, including a peak-by-cell count matrix, a quality control (QC) metrics report, and genome browser-compatible track files. This protocol focuses specifically on the data processing pipeline. Its workflow encompasses 7 key modules: sample demultiplexing, adapter sequence removal, read alignment to a reference genome, chromatin accessibility peak detection, cell barcode identification, genomic track file generation, and comprehensive quality assessment.

2.1 scATAC-pro installation

- Compile from source code on GitHub.
- Run using a pre-configured Docker image.
- Install required dependencies (Python ≥3.7, R ≥4.0, Bowtie2, etc).

Execute the following command in your terminal to deploy scATAC-pro to the specified installation directory: YOUR_INSTALL_PATH/scATAC-pro_1.5.2

```
>git clone https://github.com/wbaopaul/scATAC-
 pro.git
>cd scATAC-pro
>make configure prefix=YOUR_INSTALL_PATH
>make install
```

2.2 Input requirements

- FASTQ files for pair-end1 reads(pe1_fastq.gz), pair-end2 reads(pe2_fastq_gz) and cell barcords (index_fastq.gz)
- for data generated by 10x, you can just specify the path to each FASTQ file folder per sample

2.3 Step-by-step guide to running scATAC-pro

Execute the scATAC-pro pipeline sequentially by configuring the PEAK_CALLER (eg, MACS2) and CELL_CALLER (eg, FILTER) parameters in the configure_user.txt file prior to runtime.

```
>scATAC-pro -s demplx_fastq
-i pe1_fastq.gz,pe2_fastq.gz,index_fastq.gz
-c configure_user.txt

# or for 10x data
>scATAC-pro -s demplx_fastq
    -i pbmc_10x_fastqs/
    -c configure_user.txt

>scATAC-pro -s trimming
-i output/demplxed_fastq/pbmc5k.demplxed.PE1.
   fastq.gz,
    output/demplxed_fastq/pbmc5k.demplxed.PE2.
    fastq.gz
-c configure_user.txt

>scATAC-pro -s mapping
-i output/trimmed_fastq/pbmc5k.trimmed.demplxed.
   PE1.fastq.gz,
    output/trimmed_fastq/pbmc5k.trimmed.dem-
    plxed.PE2.fastq.gz,
-c configure_user.txt

>scATAC-pro -s call_peak
-i output/mapping_result/pbmc5k.positionsort.
   MAPQ30.bam
-c configure_user.txt

>scATAC-pro -s aggr_signal
-i output/mapping_result/pbmc5k.positionsort.
   MAPQ30.bam
-c configure_user.txt

>scATAC-pro -s get_mtx
-i output/summary/pbmc5k.fragments.tsv.gz,out-
   put/peaks/PEAK_CALLER/pbmc5k_features_
   BlacklistRemoved.bed
-c configure_user.txt

>scATAC-pro -s qc_per_barcode
-i output/summary/pbmc5k.fragments.tsv.gz,out-
   put/peaks/PEAK_CALLER/pbmc5k_features_
   BlacklistRemoved.bed
-c configure_user.txt

>scATAC-pro -s call_cell
-i output/raw_matrix/PEAK_CALLER/matrix.mtx
-c configure_user.txt

>scATAC-pro -s get_bam4Cells
-i output/mapping_result/pbmc5k.positionsort.
   bam,
output/filtered_matrix/PEAK_CALLER/CELL_CALLER/
barcodes.txt
-c configure_user.txt
```

Some of the processing modules can be run together by a single command:

In one mode, processing data including demplx_fastq, mapping, call_peak, get_mtx, aggr_signal, qc_per_barcode, call_cell, and get_bam4Cells.

```
>scATAC-pro -s process
-i pe1.fastq.gz,pe2.fastq.gz,index.fastq.gz
-c configure_user.txt

# or for 10x data
>scATAC-pro -s process
    -i PATH_TO_10x_fastqs_directory
    -c configure_user.txt
```

In another mode, conduct all processing modules except demultiplexing step.

```
>scATAC-pro -s process_no_dex
    -i pe1_fastq,pe2_fastq
    -c configure_user.txt
```

Step 3: Analysis of single-cell chromatin accessibility data with ArchR

The first thing we do is change to our desired working directory, set the number of threads we would like to use, and load

our gene and genome annotations. Then, using ArchR, we import the fragment files from the output of Cell Ranger ATAC to create ArrowFiles and an ArchRProject.

3.1 Load ArchR

```
>library(ArchR)
>library(parallel)
>library(ggplot2)
>inputFiles <- getTutorialData("Hematopoiesis")
>set.seed(123)
>addArchRThreads(threads = 16)
# We recommend setting threads to 1/2 to 3/4 of
the total available cores.
>addArchRGenome("hg38")
# addArchRGenome("hg19")
```

Note: ArchR has built-in support for "hg19," "hg38," "mm9," and "mm10." You can also use the createGeneAnnotation() and createGenomeAnnotation() functions to build your own genome and gene annotations.

3.2 Create ArrowFiles

As a powerful tool for single-cell ATAC-seq data analysis, ArchR is compatible with diverse data input formats, with fragment and BAM files being the most commonly used. Subsequently, we employ the fragment files generated by the 10x Genomics Cell Ranger ATAC as input files in ArchR.

```
# Locate Fragment Files
>inputFiles <- c("Sample1" = "/path/to/atac_
pbmc_5k_analysis/outs/fragments.tsv.gz")

# Create Arrow Files
>ArrowFiles <- createArrowFiles(
  inputFiles = inputFiles,
  sampleNames = names(inputFiles),
  minTSS = 2,
  # Dont set this too high because you can
  always increase later
  minFrags = 1000,
  addTileMat = TRUE,
  addGeneScoreMat = TRUE)
```

We can inspect the ArrowFiles object to see that it is actually just a character vector of Arrow file paths.

3.3 Per-cell quality control

In ArchR, QC of scATAC-seq data is crucial for eliminating low-quality cells. This process focuses on 3 key aspects: the number of unique nuclear fragments, the signal-to-background ratio, and the fragment size distribution. In the prior step, we applied relaxed thresholds for the minimum transcription start site (minTSS) enrichment score and the minimum mapped ATAC-seq fragments per cell (minFrags). This allowed an initial evaluation of the data quality. For each sample, ArchR automatically generates a cell density heatmap of TSS enrichment score and number of ATAC-seq fragments. Arrow files creation produces a "QualityControl" folder in the current directory, containing 2 plots per sample. One plots log10(unique nuclear fragments) against TSS enrichment scores, with thresholds marked by dotted lines (**Fig. 2A**). The other displays fragment size distributions (**Fig. 2B**).

3.4 Doublet Inference with ArchR

Single-cell data from any platform is prone to doublets, where one droplet contains one barcoded bead and multiple nuclei. This makes reads from multiple cells appear as a single cell. It is especially problematic in developmental/trajectory data, as doublets can be mistaken for intermediate cell types or states. We can observe that the addition of doublet scores leads to the generation of 3 related plots: doublet enrichments (**Fig. 2C**), doublet scores (**Fig. 2D**), and doublet density (**Fig. 2E**).

```
> doubScores <- addDoubletScores(
input = ArrowFiles,
k = 10,
#Refers to how many cells near a "pseudo-doublet"
to count.
knnMethod = "UMAP,"
#Refers to the embedding to use for nearest
neighbor search with doublet projection.
LSIMethod = 1)
```

3.5 Create an ArchRProject

An ArchRProject serves as a centralized structure for integrating multiple Arrow files into a single, memory-efficient project. It enables rapid data access and manipulation, facilitating efficient interaction with Arrow files. As the foundation of most ArchR functions and analytical workflows, it streamlines data management and accelerates computational processes.

```
>archr_proj <- ArchRProject(
  ArrowFiles = ArrowFiles,
  outputDirectory = " output_ArchR,"
  copyArrows = TRUE
  # It keeps an original Arrow file copy for
  later use, as we'll modify them downstream.)

>getAvailableMatrices(archr_proj)
```

Plotting sample fragment size distribution and TSS enrichment profiles.

```
# Fragment size distributions
>p1 <- plotFragmentSizes(ArchRProj = archr_proj)

# TSS enrichment profiles
>p2 <- plotTSSEnrichment(ArchRProj = archr_proj)
>plotPDF(p1,p2, name = "QC-Sample-FragSizes-
TSSProfile.pdf," ArchRProj = archr_proj, addDOC
= FALSE, width = 5, height = 5)
```

We can save our archr_proj using saveArchRProject(), which will allow us to use this project in the future.

```
> # save this project
>archr_proj <- saveArchRProject(
    ArchRProj = archr_proj,
    outputDirectory = "/Path_to_YourProject/
    archr_proj,"
    load = TRUE
    )

# load this project
>archr_proj <- loadArchRProject(
path = "/Path_to_YourProject/archr_proj,"
force = FALSE)
```

3.6 Filtering doublets from an ArchRProject

A key part of this step is filterRatio, which is the maximum ratio of predicted doublets to filter, based on the number of pass—filter cells.

```
>archr_proj1 <- filterDoublets(archr_proj)
>archr_proj1 <- archr_proj1[which(archr_proj1$TS-
SEnrichment > 5 & archr_proj1$nFrags < 50000)]
```

3.7 Uniform Manifold Approximation and Projection (UMAP) of dimensionality reduction

Dimensionality reduction in scATAC-seq is challenging due to the sparsity of the data, as most loci have zero accessible alleles, making traditional methods like PCA ineffective. To address this, latent semantic indexing (LSI),[23,24] originally from natural language processing, is used. LSI normalizes data using term frequency-inverse document frequency (TF-IDF) and applies singular value decomposition (SVD) to reduce dimensionality while retaining important features. This transformation allows for effective visualization using UMAP or t-SNE, referred to as embeddings in ArchR.

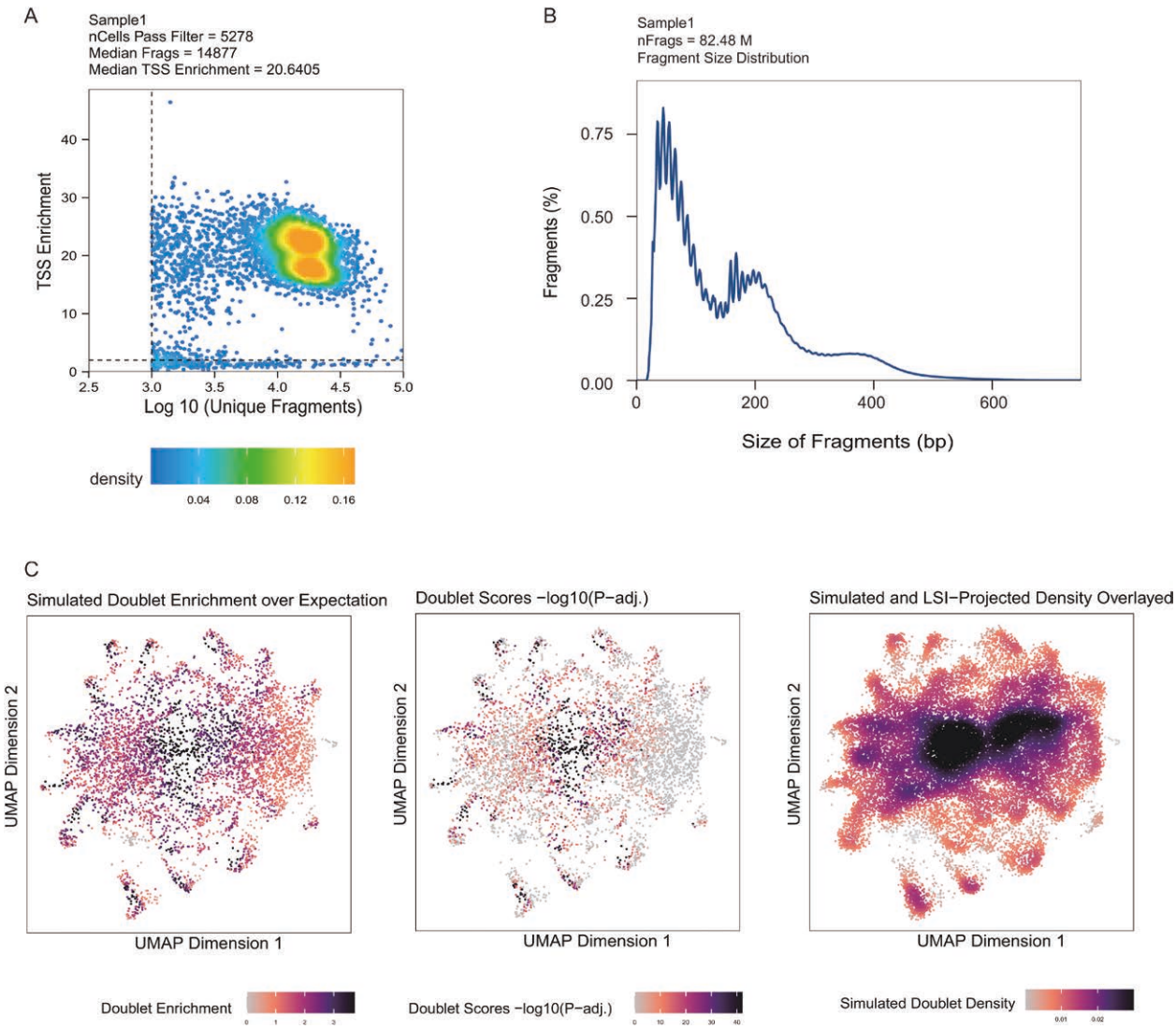

Figure 2. Quality control and doublet detection of scATAC-seq data. (A–B) Quality control metrics of sample 1. (A) TSS enrichment score versus $\log_{10}$(unique nuclear fragments) per cell. Dotted lines represent filtering thresholds applied for downstream analysis. Cells passing these thresholds (n = 5278) exhibited a median of 14,877 fragments and a median TSS enrichment score of 20.64, indicating good data quality. Fragment size distribution (B) showing characteristic mono- and di-nucleosome peaks, supporting successful transposition and library construction. (C) Doublet identification plots. UMAP projection showing doublet enrichment scores (left), indicating the relative enrichment of simulated doublets compared to expectation. UMAP projection of doublet scores (middle), reflecting the statistical significance ($-\log_{10}$ binomial-adjusted *p* value) of doublet enrichment for each cell. UMAP projection of simulated doublet density (right), visualizing the spatial distribution of synthetic doublets projected into the same low-dimensional embedding. These plots were generated during the addDoubletScores() step and are used to identify and filter potential doublets. LSI = latent semantic indexing, scATAC = single-cell assay for transposase-accessible chromatin, TSS = transcription start site, UMAP = Uniform Manifold Approximation and Projection.

```
>archr_proj1 <- addIterativeLSI(
ArchRProj = archr_proj1,
useMatrix = "TileMatrix,"
name = "IterativeLSI,"
iterations = 2,
clusterParams = list(#See Seurat::FindClusters
    resolution = c(0.2),
    sampleCells = 5000,
    n.start = 10),
varFeatures = 25000,
dimsToUse = 1:30)
```

3.8 Clustering with ArchR

```
# Clustering using Seurat's `FindClusters()` function
>archr_proj1 <- addClusters(
    input = archr_proj1,
    reducedDims = "IterativeLSI",
    method = "Seurat",
    name = "Clusters",
    resolution = 0.05)
```

3.9 Single-cell embeddings

We use the addUMAP() function to run UMAP in ArchR and divide these cells into 5 clusters(**Fig. 3A**).

```
>archr_proj1 <- addUMAP(
  ArchRProj = archr_proj1,
  reducedDims = "IterativeLSI,"
  name = "UMAP,"
  nNeighbors = 30,
  minDist = 0.5,
  metric = "cosine")

>archr_proj1@embeddings
>p3 <- plotEmbedding(archr_proj1, colorBy = "cell-
 ColData," name = "Clusters," embedding = "UMAP")
```

3.10 Batch effect correction with Harmony

If there are subtle batch effects, you can add more LSI iterations and start with a lower initial clustering resolution. For

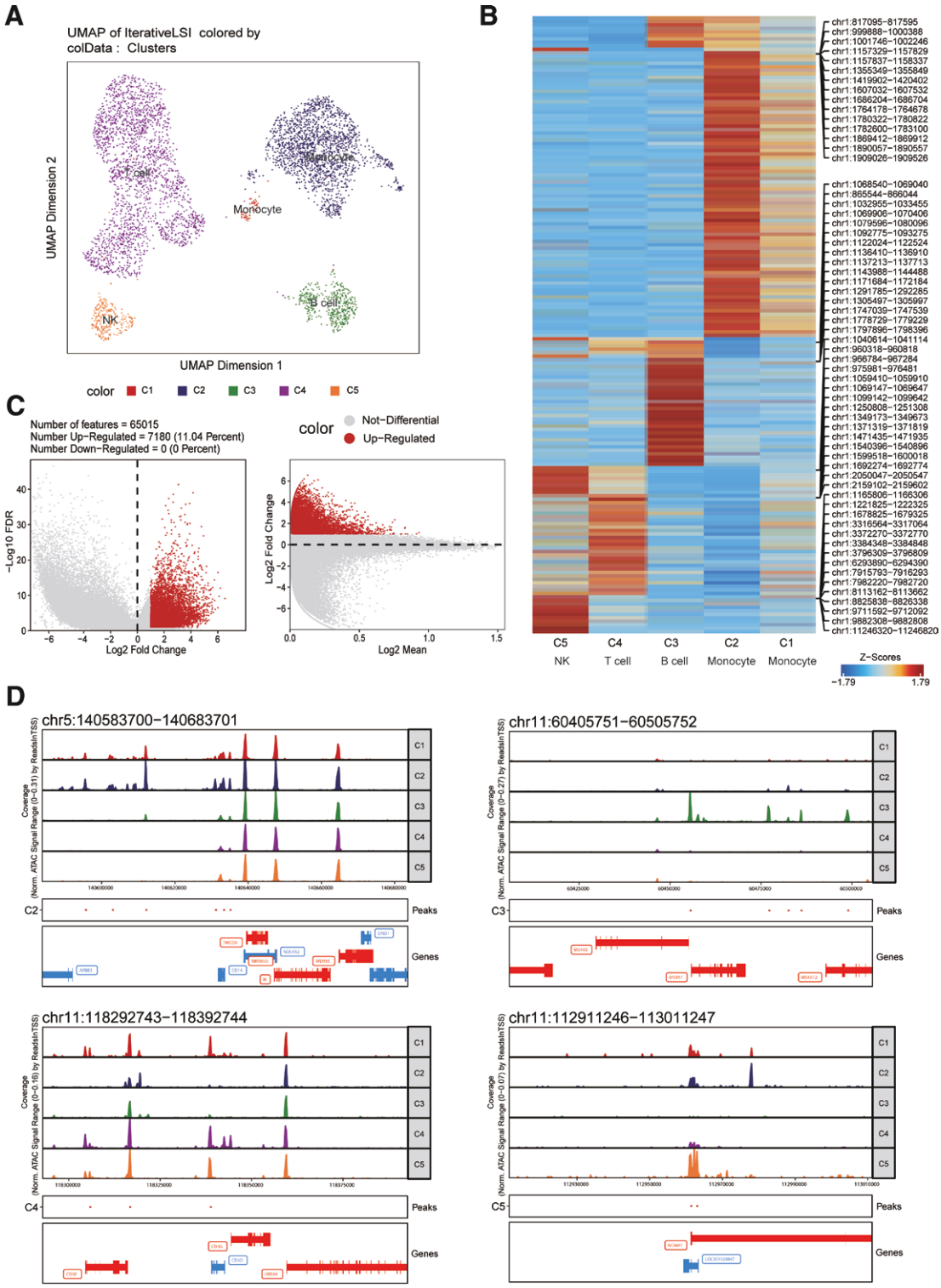

Figure 3. Chromatin accessibility landscape and differential peak analysis across identified cell clusters. (A) UMAP embedding based on Iterative LSI showing 5 distinct clusters (C1–C5) identified from dataset. Each dot represents a single cell, colored by its cluster assignment. (B) Heatmap of top marker peaks across clusters (C1–C5), generated by markerHeatmap. Accessibility scores are *Z* score normalized, highlighting cluster-specific chromatin signatures. Genomic coordinates of selected regions are shown on the y-axis. (C) MA plot (right) and volcano plot (left) of differentially accessible chromatin regions for cluster C4, generated using plotMarkers() from ArchR. Peaks were selected using thresholds of FDR ≤0.1 and $\log_2$ fold change ≥1. Red points represent significantly upregulated peaks in C4 relative to other clusters. (D) Genome browser tracks showing chromatin accessibility at selected marker loci across 5 cell clusters (C1–C5). ATAC-seq signal tracks are grouped by clusters, with significantly upregulated peaks (FDR ≤0.1, Log_2FC ≥1) highlighted at each locus. ATAC = assay for transposase-accessible chromatin, FDR = false discovery rate, LSI = latent semantic indexing, MA = mean-average, UMAP = Uniform Manifold Approximation and Projection.

strong batch effect differences, use ArchR's Harmony tool for correction.

```
>archr_proj1 <- addHarmony(
  ArchRProj = archr_proj1,
  reducedDims = "IterativeLSI,"
  name = "Harmony,"
  groupBy = "Sample")

>archr_proj1 <- addClusters(
  input = archr_proj1,
  reducedDims = "Harmony,"
  method = "Seurat,"
  name = "Harmony_Clusters,"
  maxClasters = 35,
  resolution = 0.8)

>archr_proj1 <- addUMAP(
  ArchRProj = archr_proj1,
  reducedDims = "Harmony,"
  name = "UMAPHarmony,"
  nNeighbors = 30,
  minDist = 0.5,
  metric = "cosine")
```

Beyond our chosen approach, several popular batch correction and integration methods have been evaluated for scATAC-seq data, including LIGER,[25] Harmony,[26] RPCA,[27] BBKNN,[28] and scVI.[29] These tools differ in their underlying assumptions and performance across various feature spaces such as peaks, genomic windows, or gene activity matrices. Notably, only a minority (~27%) of integration attempts across diverse mouse brain datasets yielded improvements over unintegrated data, emphasizing the importance of feature selection.[30] LIGER showed robust batch correction across peak and window spaces, though sometimes introduced artificial substructures. Harmony retained more biological variation but exhibited occasional subtype overlaps. RPCA and BBKNN offered a balance between correction and signal retention in small-scale datasets, though with limited scalability.

Overall, when integrating scATAC-seq datasets, careful selection of feature space and tool is essential, as current methods often struggle to simultaneously remove batch effects and preserve biological fidelity—particularly in high-dimensional contexts.

3.11 Defining cluster identity

Through manual annotation, we defined the *5* clusters as 4 cell types.

```
>markersGS <- getMarkerFeatures(
  ArchRProj = archr_proj1,
  useMatrix = "GeneScoreMatrix,"
  groupBy = "Clusters,"
  bias = c("TSSEnrichment," "log10(nFrags)"),
  testMethod = "wilcoxon")

>markerList <- getMarkers(markersGS, cutOff =
"FDR≤0.05 & Log2FC≥1.0")

>markerGenes <- c(
  "CD34," #Early Progenitor
  "GATA1," #Erythroid
  "CD79A," "MS4A1," "CD19," #B-Cells
  "CD14," "MPO," #Monocytes
  "CD3D," "CD8A,"#TCells
  "NCAM1" #NK)

>p4 <- plotEmbedding(
  ArchRProj = archr_proj1,
  colorBy = "GeneScoreMatrix,"
  name = markerGenes,
  embedding = "UMAP,"
  quantCut = c(0.01, 0.95),
  imputeWeights = NULL)

>cluster_annotations <- c(
  "C1" = "Monocyte,"
  "C2" = "Monocyte,"
  "C3" = "B Cell,"
  "C4" = "T Cell,"
  "C5" = "NK Cell")

>current_clusters <- archr_proj1$Clusters
>archr_proj1$celltype    <-    cluster_annotations[as.character(current_clusters)]

>p5 <- plotEmbedding(archr_proj1, colorBy = "cellColData," name = "celltype," embedding = "UMAP")
```

Step 4: Calling peaks with ArchR

Before calling peaks in ArchR, you must first run addGroupCoverages(), as ArchR relies on these group coverage objects for peak calling.

```
# Making Pseudo-bulk Replicates

> archr_proj <- addGroupCoverages(ArchRProj =
archr_proj1, groupBy = "Clusters")
```

Peak calling was performed using MACS2 via ArchR's "addReproduciblePeakSet()" function, with the parameters "--nomodel --shift -100 --extsize 200 --qvalue 0.05." This configuration is recommended for scATAC-seq data, as it accounts for Tn5 transposase insertion sites and avoids model mis-specification in sparse single-cell data. Although ArchR includes a built-in peak caller, MACS2 is preferred due to its improved sensitivity and reproducibility.

To ensure the robustness of peak calling, we evaluated 3 representative MACS2 parameter sets that varied in "—qvalue" and "—extsize." All configurations yielded the same number of final union peaks (65,015) after ArchR's reproducibility filtering, with only 8 peaks overlapping ENCODE blacklist regions in each case (<0.01%). These results indicate that peak detection in our dataset is highly stable and minimally affected by small changes in parameterization.

For successful integration of MACS2, ArchR locates the executable through the system PATH or pip installation. If not automatically detected, the user can specify the exact binary location via the "pathToMacs2" argument. We recommend verifying the MACS2 version being used to avoid conflicts in multi-environment setups.

```
# pathToMacs2 <- findMacs2()
>pathToMacs2 <- normalizePath("/usr/local/bin/
macs2")

>archr_proj2 <- addReproduciblePeakSet(
  ArchRProj = archr_proj2,
  groupBy = "Clusters,"
  pathToMacs2 = pathToMacs2,
  peakMethod = "MACS2,"
  additionalParams = "--nomodel --shift -100
--extsize 200 --qvalue 0.05")

>getPeakSet(archr_proj2)

# Adding a Peak Matrix
>archr_proj3 <- addPeakMatrix(archr_proj2)
```

Step 5: Differential chromatin accessibility peaks analysis

In many cases, identifying peaks that are unique to a specific cluster or a small group of clusters is crucial. In ArchR, this can be achieved in an unsupervised manner by utilizing the

addMarkerFeatures() function in combination with the useMatrix = "PeakMatrix" parameter. This approach allows for the detection of cluster-specific chromatin accessibility features, providing valuable insights into the distinct regulatory landscapes of individual clusters.

```
>markersPeaks <- getMarkerFeatures(
  ArchRProj = archr_proj3,
  useMatrix = "PeakMatrix,"
  groupBy = "Clusters,"
  bias = c("TSSEnrichment," "log10(nFrags)"),
  testMethod = "wilcoxon")


>markerPeaksList  <-  getMarkers(markersPeaks,
 cutOff = "FDR≤0.05 & Log2FC≥1.0")
```

ArchR provides multiple plotting functions that interface with objects returned by SummarizedExperiment, enabling visualization of marker peaks within the ArchR framework(**Fig. 3B–D**).

```
# Marker Peak Heatmaps
>heatmapPeaks <- markerHeatmap(
    seMarker = markersPeaks,
    cutOff = "FDR≤0.1 & Log2FC≥0.5,"
    transpose = TRUE)


>draw(heatmapPeaks,    heatmap_legend_side    =
 "bot," annotation_legend_side = "bot")


# Marker Peak MA and Volcano Plots
>pma  <-  plotMarkers(seMarker  =  markersPeaks,
name  =  "C4," cutOff  =  "FDR≤0.1  &  Log2FC≥1,"
plotAs = "MA")
>pv  <-  plotMarkers(seMarker  =  markersPeaks,
name  =  "C4," cutOff  =  "FDR≤0.1  &  Log2FC≥1,"
plotAs = "Volcano")
>plotPDF(pma,  pv,  name  =  "Tcell-Markers-MA-
 Volcano.pdf," width = 5, height = 5, ArchRProj
 = archr_proj3, addDOC = FALSE)


# Marker Peaks in Browser Tracks
# C4--CD3D--T cell
>p6 <- plotBrowserTrack(
    ArchRProj = archr_proj3,
    groupBy = "Clusters,"
    geneSymbol = c("CD3D"),
    0features = getMarkers(markersPeaks, cutOff =
    "FDR≤0.1 & Log2FC≥1," returnGR = TRUE)["C4"],
    upstream = 50000,
    downstream = 50000)
>grid::grid.newpage()
>grid::grid.draw(p6$CD3D)
```

Step 6: Motif and feature enrichment analysis with ArchR

After identifying marker peaks, we can perform motif enrichment analysis on these peaks using the peakAnnoEnrichment() function. Marker peaks are typically derived from significantly differentially accessible peaks, identified through the getMarkerFeatures() function in ArchR. By using peakAnnoEnrichment(), we test whether these peaks are enriched for specific transcription factor (TF) binding motifs, helping to uncover potential TF regulatory relationships(**Fig. 4A**). This step provides critical insights into the gene regulatory mechanisms within different cell populations.

```
> archr_proj3 <- addMotifAnnotations(ArchRProj =
  archr_proj3, motifSet = "cisbp," name = "Motif")


# Motif Enrichment in Marker Peaks
>enrichMotifs <- peakAnnoEnrichment(
    seMarker = markersPeaks,
    ArchRProj = archr_proj3,
    peakAnnotation = "Motif,"
    cutOff = "FDR≤0.1 & Log2FC≥0.5")
>heatmapEM  <-  plotEnrichHeatmap(enrichMotifs,
 n = 7, transpose = TRUE)


>ComplexHeatmap::draw(heatmapEM,  heatmap_leg-
end_side  =  "bot,"  annotation_legend_side  =
"bot")
```

Step 7: ChromVAR deviations enrichment

ChromVAR is an R package designed to analyze sparse single-cell ATAC-seq data by quantifying variability in chromatin accessibility across peaks sharing the same motif or annotation while correcting for technical biases.[20] In ArchR, this is achieved by first using chromVAR::getBackgroundPeaks() to define a set of matched background peaks, and then applying the addDeviationsMatrix() function to compute motif deviation scores for each cell across all annotated motifs. We can plot variable deviations (**Fig. 4B**). This process generates a "MotifMatrix" that captures chromatin accessibility variation associated with TF binding. Through the names of the features we are interested in, we have plotted the distribution of chromVAR deviation scores for each cluster (**Fig. 4C**).

```
> if("Motif" %ni% names(archr_proj3@peakAnno-
tation)){
archr_proj3    <-    addMotifAnnotations(ArchR-
Proj = archr_proj3, motifSet = "cisbp," name =
"Motif")}

>archr_proj3 <- addBgdPeaks(archr_proj3)
>archr_proj3 <- addDeviationsMatrix(
    ArchRProj = archr_proj3,
    peakAnnotation = "Motif,"
    force = TRUE)

>plotVarDev  <-  getVarDeviations(archr_proj3,
name = "MotifMatrix," plot = TRUE)

# extract a subset of motifs for downstream analysis
>motifs <- c("GATA1," "CEBPA," "EBF1," "IRF4,"
"TBX21," "PAX5," "Eomes," "Nfil3","LEF1," "TCF7")
>markerMotifs    <-    getFeatures(archr_proj3,
select = paste(motifs, collapse="|"), useMatrix
= "MotifMatrix")
>markerMotifs <- grep("z:," markerMotifs, value
= TRUE)
>markerMotifs <- markerMotifs[markerMotifs %ni%
c("z:SREBF1_22","z:TCF7L1_763,""z:TCF7L2_762")]

>plotGroups(ArchRProj = archr_proj3,
  groupBy = "Clusters,"
  colorBy = "MotifMatrix,"
  name = markerMotifs,
  imputeWeights=getImputeWeights(archr_proj3))
```

We can visualize the distribution of chromVAR deviation scores across each cluster and overlay the corresponding TF *z* scores onto the UMAP embedding. Additionally, gene scores for these TF can also be overlaid on the UMAP to provide a comprehensive view of their activity patterns across different cell populations.

```
# overlay the z-scores on UMAP embedding
>plotEmbedding(
    ArchRProj = archr_proj3,
    colorBy = "MotifMatrix,"
    name = sort(markerMotifs),
    embedding = "UMAP,"
    imputeWeights                                =
getImputeWeights(archr_proj3))

# overlay the GeneScores for each of TFs on the
UMAP embedding
>markerGS  <-  getFeatures(archr_proj3,  select
=  paste(motifs,  collapse="|"),  useMatrix  =
"GeneScoreMatrix")
>markerGS <- markerGS[markerGS %ni% c("SREBF1",
"CEBPA-DT,""TCF7L2," "TCF7L1," "LEF1-AS1")]
>plotEmbedding(
```

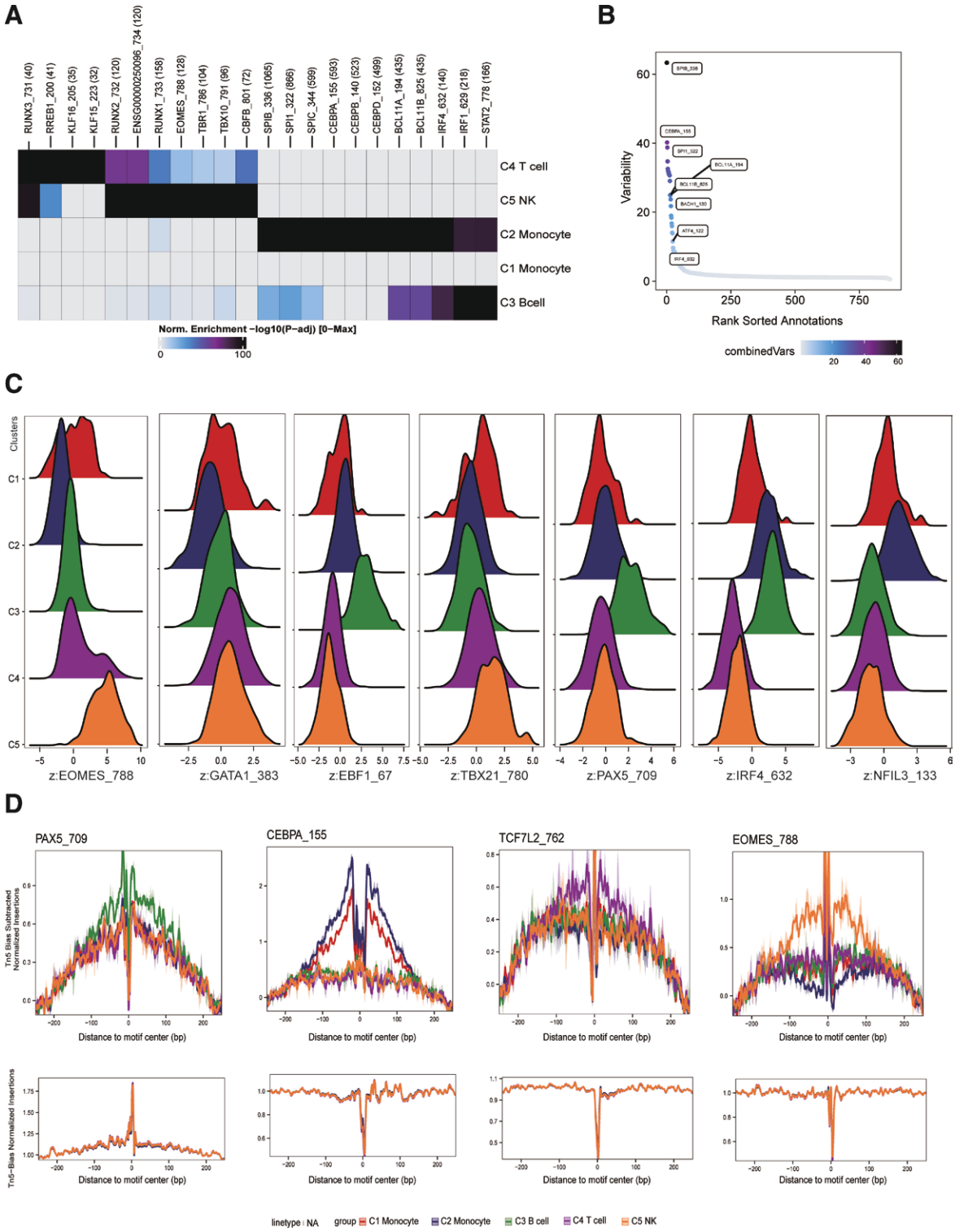

Figure 4. TF motif enrichment and footprinting analysis across scATAC-seq-defined clusters. (A) Heatmap showing TF motif enrichment in differentially accessible peaks across cell clusters. Motifs from the CIS-BP database were annotated via addMotifAnnotations(), and significance is represented as normalized enrichment scores (color gradient) and adjusted *p* values (–log10[p adj]). Each row represents a transcription factor motif and each column a cell cluster. (B) Rank-sorted motif variability plot showing TF motifs with the highest variability across 5 cell clusters (C1–C5). Each dot represents a TF motif, ranked by variability on the y-axis, which is calculated as the standard deviation of *z* scores across all cells. Variability reflects cell-to-cell heterogeneity in motif accessibility. Highly variable motifs are labeled and represent potential regulators of cell type-specific chromatin landscapes. (C) ChromVAR deviation score distributions for selected TF motifs across 5 cell clusters. The ridge plot shows smoothed deviation *z* scores for 7 immune-related TF motifs across clusters. Higher scores indicate greater motif accessibility. Imputation was applied to account for data sparsity in scATAC-seq. (D) TF footprinting using ArchR with Tn5 bias subtraction. Footprinting analysis was performed on selected TF motifs (PAX5_709, CEBPA_155, TCF7L2_762, and EOMES_788) using pseudo-bulk chromatin accessibility data grouped by clusters. scATAC = single-cell assay for transposase-accessible chromatin, TF = transcription factor.

```
ArchRProj = archr_proj3,
    colorBy = "GeneScoreMatrix,"
    name = sort(markerGS),
    embedding = "UMAP,"
    imputeWeights=getImputeWeights(archr_proj3))
```

Step 8: TF motif annotation and footprinting analysis in ArchR

TF footprinting analysis predicts the precise binding sites of TFs on the chromatin, which typically requires high sequencing depth to achieve accurate results. To address this limitation, an integrative analytical strategy is implemented by systematically correlating predicted TF binding sites with the spatial distribution patterns of Tn5 transposase insertion sites. The accuracy of this footprinting analysis heavily depends on the availability of a reliable list of predicted binding sites for the TF of interest.

Motif annotation in the ArchR workflow is performed using the addMotifAnnotations() function, which applies sequence matching algorithms to scan chromatin-accessible regions (peaks) for DNA sequences matching the target TF motifs. The results are stored in the ArchRProject object as binary annotations (0 = motif absent, 1 = motif present), indicating the presence or absence of the motif in each peak. While this approach efficiently identifies canonical motif elements, its accuracy can be limited by motif degeneracy, potentially overlooking non-canonical or low-affinity binding sites.

Once motif annotations are generated, TF footprinting analysis is conducted using the getFootprints() function, which requires 2 inputs: the ArchRProject object and a set of motif coordinates in GenomicRanges format, derived from the getPositions() function. This procedure quantifies protection patterns at predicted TF binding sites, inferred from transposase (Tn5) insertion profiles, thus enabling the detection of TF occupancy at single-base resolution.

This analysis framework is part of the validated ArchR pipeline, which provides scalable and efficient single-cell chromatin accessibility analysis, particularly suited for high-throughput scATAC-seq datasets.[11] The binary motif annotation structure, compatible with BSgenome, ensures computational efficiency for large-scale datasets and delivers structured inputs for downstream inference of GRNs.

```
>motifPositions <- getPositions(archr_proj3)
>motifs <- c("GATA1," "CEBPA," "EBF1," "IRF4,"
 "TBX21," "PAX5")
>markerMotifs <- unlist(lapply(motifs, function(x)
 grep(x, names(motifPositions), value = TRUE)))
>markerMotifs   <-   markerMotifs[markerMotifs
 %ni% "SREBF1_22"]

>if(is.null(archr_proj3@projectMetadata$Group-
 Coverages$Clusters)){archr_proj3 <- addGroup-
 Coverages(ArchRProj = archr_proj3, groupBy =
 "Clusters")}

>seFoot    <-    getFootprints(ArchRProj    =
archr_proj3,
  positions = motifPositions[markerMotifs],
  groupBy = "Clusters")
```

The resulting footprints can be visualized using the plotFootprints() function, which displays the distribution of motif insertions, reflecting the binding activity of the TFs.

To eliminate technical noise and reduce false-positive signals introduced by the sequence preference of Tn5 transposase, normalization for Tn5 bias is essential in footprinting analysis. This correction improves the specificity of footprint detection and ensures the reliability of downstream analyses. In this study, we applied 2 normalization strategies implemented in ArchR: subtracting Tn5 (**Fig. 4D**) bias and dividing by Tn5 bias, each providing a distinct approach to account for bias-related artifacts in Tn5 insertion patterns.

```
# Subtracting the Tn5 Bias
>plotFootprints(
    seFoot = seFoot,
    ArchRProj = archr_proj3,
    normMethod = "Subtract,"
    plotName = "Footprints-Subtract-Bias,"
    addDOC = FALSE,
    smoothWindow = 5)


# Dividing by the Tn5 Bias
>plotFootprints(
    seFoot = seFoot,
    ArchRProj = archr_proj3,
    normMethod = "Divide,"
    plotName = "Footprints-Divide-Bias,"
    addDOC = FALSE,
    smoothWindow = 5)


# Footprinting Without Normalization for Tn5
Bias
>plotFootprints(
    seFoot = seFoot,
    ArchRProj = archr_proj3,
    normMethod = "None,"
    plotName = "Footprints-No-Normalization,"
    addDOC = FALSE,
    smoothWindow = 5)
```

Step 9: Transcriptional regulatory network construction with SCENIC+

Here, we employ SCENIC+, a comprehensive framework that integrates single-cell chromatin accessibility, gene expression, and TF motif information to reconstruct enhancer-driven gene regulatory networks (eGRNs).[21]

The resulting network allows high-resolution exploration of transcriptional regulation.

While early steps of the chromatin accessibility analysis (eg, peak calling, clustering, and motif enrichment) were performed on an ATAC-only dataset, SCENIC+ analysis was conducted using a paired single-cell multiome dataset (RNA + ATAC) from 10x Genomics. This ensured compatibility with SCENIC+'s multi-modal regulatory network reconstruction framework.

9.1 Install SCENIC+ and Dependencies

Installing SCENIC+ in the new conda environment is highly recommended. It can be installed in a conda environment as follows:

```
>conda create --name scenicplus python=3.11 -y
>conda activate scenicplus
>git clone https://github.com/aertslab/scenicplus
>cd scenicplus
>pip install.
```

9.2 Preprocess scATAC-seq Data using pycisTopic

SCENIC+ enables the integration of ArchR-filtered fragment count matrices and cell metadata into a cisTopic object, facilitating subsequent topic modeling and GRN inference. Additionally, predefined region sets can be used to bypass topic modeling, though this may limit the resolution of regulatory pattern discovery. The flexibility in input formats facilitates integration with existing analysis pipelines.

We begin by creating a cisTopic object, then add metadata to the cisTopic object, and subsequently infer doublets from the scATAC-seq data.

```
>from pycisTopic.cistopic_class import create_
cistopic_object_from_fragments
>import polars as pl
>cistopic_obj_list = []
>for sample_id in fragments_dict:
    sample_metrics = pl.read_parquet(
      os.path.join(pycistopic_qc_output_dir, f'
{sample_id}.fragments_stats_per_cb.parquet')
    ).to_pandas().set_index("
```

```
CB"
).loc[sample_id_to_barcodes_passing_fil-
ters[sample_id]]
    cistopic_obj = create_cistopic_object_from_
fragments(
    path_to_fragments                          =
fragments_dict[sample_id],
    path_to_regions = path_to_regions,
    path_to_blacklist = path_to_blacklist,
    metrics = sample_metrics,
    valid_bc = sample_id_to_barcodes_passing_fil-
ters[sample_id],
    n_cpu = 1,
    project = sample_id,
    split_pattern = '-')
>cistopic_obj_list.append(cistopic_obj)
>cistopic_obj = cistopic_obj_list[0]
>import pandas as pd
>cell_data = pd.read_table("
/home/Scenicplus_example/pbmc_tutorial/data/
cell_data.tsv", index_col = 0)
>cistopic_obj.add_cell_data(cell_data,
split_pattern='-')
>import scrublet as scr
>scrub = scr.Scrublet(cistopic_obj.fragment_
matrix.T, expected_doublet_rate=0.1)
>doublet_scores, predicted_doublets = scrub.
scrub_doublets()
>import matplotlib.pyplot as plt
>scrublet = pd.DataFrame([scrub.doublet_scores_
obs_, scrub.predicted_doublets_], columns=cistopic_
obj.cell_names, index=['Doublet_scores_fragments',
'Predicted_doublets_fragments']).T
>cistopic_obj.add_cell_data(scrublet, split_
pattern = '-')
>singlets = cistopic_obj.cell_data[cistopic_
obj.cell_data.Predicted_doublets_fragments ==
False].index.tolist()
>cistopic_obj_noDBL = cistopic_obj.subset(sin-
glets, copy=True, split_pattern='-')
```

Next, we will run topic modeling using Latent Dirichlet Allocation (LDA) with a Collapsed Gibbs Sampler and perform model selection.

```
>os.environ["MALLET_MEMORY"] = "32G"
>from pycisTopic.lda_models import
run_cgs_models_mallet
>mallet_path="/home/.conda/envs/scenicplus/lib/
python3.11/site-packages/pycisTopic/mallet"
>models=run_cgs_models(cistopic_obj,
     n_topics=[2,5,10,15,20,25],
     n_cpu=6,
     n_iter=100,
     random_state=555,
     alpha=50,
     alpha_by_topic=True,
     eta=0.1,
     eta_by_topic=False,
     save_path="/home/Scenicplus_example/
tutorial/,")
>from pycisTopic.lda_models import
evaluate_models
>model = evaluate_models(models, select_model = 20,
     return_model = True)
>cistopic_obj.add_LDA_model(model)
```

Clustering and visualization

```
>from pycisTopic.clust_vis import (
  find_clusters,
  run_umap,
  run_tsne,
  plot_metadata,
  plot_topic,
  cell_topic_heatmap)
>find_clusters(
    cistopic_obj,
    target = 'cell',
    k = 10,
    res = [0.6, 1.2, 3],
    prefix = 'pycisTopic_',
    scale = True,
split_pattern = '-')
>run_umap(cistopic_obj, target = "cell," scale=True)
>run_tsne(cistopic_obj, target = "cell," scale=True)
    >annot_dict = {}
>for resolution in [0.6, 1.2, 3]:
    annot_dict[f"pycisTopic_leiden_10_{resolu-
tion}"] = {}
    for cluster in set(cistopic_obj.cell_
    data[f"pycisTopic_leiden_10_{resolution}"]):
    counts = cistopic_obj.cell_data.loc[cis-
    topic_obj.cell_data.loc[cistopic_obj.cell_
    data[f"pycisTopic_leiden_10_{resolution}"]
    == cluster].index, "celltype"].value_counts()
    annot_dict[f"pycisTopic_leiden_10_{resolu-
    tion}"][cluster] = f"{counts.index[counts.
    argmax()]}({cluster})"
```

Topic binarization & QC

```
>from pycisTopic.topic_binarization import
binarize_topics
>region_bin_topics_top_3k = binarize_topics(
    cistopic_obj, method='ntop', ntop = 3_000,
    plot=True, num_columns=5)
>region_bin_topics_otsu = binarize_topics(
    cistopic_obj, method='otsu',
    plot=True, num_columns=5)
>region_bin_topics_otsu = binarize_topics(
    cistopic_obj, method='otsu',
    plot=True, num_columns=5)
>from pycisTopic.topic_qc import compute_topic_
 metrics, plot_topic_qc, topic_annotation
>import matplotlib.pyplot as plt
>from pycisTopic.utils import fig2img
>topic_qc_metrics = compute_topic_metrics(cis-
topic_obj)
>fig_dict={}
>fig_dict["CoherenceVSAssignments"]=plot_topic_
 qc(topic_qc_metrics, var_x='Coherence',
 var_y='Log10_Assignments', var_color='Gini_
 index', plot=False, return_fig=True)
>fig_dict["AssignmentsVSCells_in_
 bin"]=plot_topic_qc(topic_qc_metrics,
 var_x='Log10_Assignments', var_y='Cells_in_
 binarized_topic', var_color='Gini_index',
 plot=False, return_fig=True)
>fig_dict["CoherenceVSCells_in_bin"]=plot_
 topic_qc(topic_qc_metrics, var_x='Coherence',
 var_y='Cells_in_binarized_topic', var_col-
 or='Gini_index', plot=False, return_fig=True)
>fig_dict["CoherenceVSRegions_in_bin"]=plot_
 topic_qc(topic_qc_metrics, var_x='Coherence',
 var_y='Regions_in_binarized_topic', var_col-
 or='Gini_index', plot=False, return_fig=True)
>fig_dict["CoherenceVSMarginal_dist"]=plot_
 topic_qc(topic_qc_metrics, var_x='Coherence',
 var_y='Marginal_topic_dist', var_color='Gini_
 index', plot=False, return_fig=True)
>fig_dict["CoherenceVSGini_index"]=plot_topic_
 qc(topic_qc_metrics, var_x='Coherence',
 var_y='Gini_index', var_color='Gini_index',
 plot=False, return_fig=True)
>topic_annot = topic_annotation(
cistopic_obj,
    annot_var='celltype',
    binarized_cell_topic=binarized_cell_topic,
    general_topic_thr = 0.2)
```

Differentially accessible regions (DARs)

```
>from pycisTopic.diff_features import (
  impute_accessibility,
  normalize_scores,
  find_highly_variable_features,
```

```
  find_diff_features)
  >import numpy as np
>imputed_acc_obj = impute_accessibility(
  cistopic_obj,
  selected_cells=None,
  selected_regions=None,
  scale_factor=10**6)
>normalized_imputed_acc_obj = normalize_
scores(imputed_acc_obj, scale_factor=10**4)
>variable_regions =
find_highly_variable_features(
  normalized_imputed_acc_obj,
  min_disp = 0.05,
  min_mean = 0.0125,
  max_mean = 3,
  max_disp = np.inf,
  n_bins=20,
  n_top_features=None,
  plot=True)
>markers_dict= find_diff_features(
  cistopic_obj,
  imputed_acc_obj,
  variable='celltype',
  var_features=variable_regions,
  contrasts=None,
  adjpval_thr=0.05,
  log2fc_thr=np.log2(1.5),
  n_cpu=5,
  _temp_dir='/home/Task/tem_dir/',
  split_pattern = '-')
```

Save region sets

```
>os.makedirs(os.path.join(out_dir, "region_
 sets"), exist_ok = True)
>os.makedirs(os.path.join(out_dir, "region_
 sets," "Topics_otsu"), exist_ok = True)
>os.makedirs(os.path.join(out_dir, "region_
 sets," "Topics_top_3k"), exist_ok = True)
>os.makedirs(os.path.join(out_dir, "region_
 sets," "DARs_cell_type"), exist_ok = True)
>from pycisTopic.utils import
region_names_to_coordinates
>for topic in region_bin_topics_otsu:
  region_names_to_coordinates(
    region_bin_topics_otsu[topic].index
    ).sort_values(
     ["Chromosome", "Start", "End"]
  ).to_csv(
  os.path.join(out_dir, "region_sets", "Topics_
  otsu", f" {topic}.bed"),
    sep = "\t",
    header = False, index = False)
>for topic in region_bin_topics_top_3k:
  region_names_to_coordinates(
    region_bin_topics_top_3k[topic].index
  ).sort_values(
     ["Chromosome", "Start", "End"]
).to_csv(
    os.path.join(out_dir, "region_sets",
    "Topics_top_3k", f" {topic}.bed"),
    sep = "\t",
    header = False, index = False)
>for cell_type in markers_dict:
    region_names_to_coordinates(
    markers_dict[cell_type].index
    ).sort_values(
     ["Chromosome", "Start", "End"]
).to_csv(
    os.path.join(out_dir, "region_sets", "DARs_
    cell_type", f" {cell_type}.bed"),
    sep = "\t",
    header = False, index = False)
```

Gene activity

```
>import pyranges as pr
>import pandas as pd
>from pycisTopic.gene_activity import
get_gene_activity
>chromsizes = pd.read_table(os.path.join(out_
 dir, "qc," "hg38.chrom_sizes_and_alias.tsv"))
>chromsizes.rename({"# ucsc": "Chromosome,"
 "length": "End"}, axis = 1, inplace = True)
>chromsizes["Start"] = 0
>chromsizes = pr.PyRanges(chromsizes[["Chromo-
some," "Start," "End"]])
>pr_annotation = pd.read_table(
    os.path.join(out_dir, "qc", "tss.bed")
    ).rename(
    {"Name": "Gene", "# Chromosome": "Chromosome"
}, axis = 1)
>pr_annotation["Transcription_Start_Site"] =
 pr_annotation["Start"]
>pr_annotation = pr.PyRanges(pr_annotation)
>gene_act, weigths = get_gene_activity(im-
 puted_acc_obj, pr_annotation, chromsizes,
 use_gene_boundaries=True, upstream=[1000,
 100000], downstream=[1000,100000], dis-
 tance_weight=True, decay_rate=1,
 extend_gene_body_upstream=10000, extend_gene_
 body_downstream=500, gene_size_weight=False,
 gene_size_scale_factor="median," remove_pro-
 moters=False, average_scores=True, scale_fac-
 tor=1, extend_tss=[10,10], gini_weight = True,
 return_weights= True, project='Gene_activity')
>DAG_markers_dict= find_diff_features(cistopic_
 obj, gene_act, variable='celltype', var_fea-
 tures=None, contrasts=None, adjpval_thr=0.05,
 log2fc_thr=np.log2(1.5), n_cpu=5, _temp_dir='/
 home/Task/tem_dir/', split_pattern = "-")
```

Label transfer

```
>from pycisTopic.label
>import scanpy as sc
>rna_anndata = sc.read_h5ad(
  "/home/Scenicplus_example/pbmc_tutorial/scRNA/
adata.h5ad"
).raw.to_adata()
>atac_anndata = sc.AnnData(gene_act.mtx.T, obs =
 pd.DataFrame(index = gene_act.cell_names), var
 = pd.DataFrame(index = gene_act.feature_names))
>atac_anndata.obs["sample_id"] = "10x_pbmc"
  >rna_anndata.obs["sample_id"] = "10x_pbmc"
  >label_dict = label_transfer(
  rna_anndata,
  atac_anndata,
  labels_to_transfer = ['celltype'],
  variable_genes = True,
  methods = ['ingest', 'harmony', 'bbknn',
'scanorama', 'cca'],
return_label_weights = False,
  _temp_dir= '/share/home/zhangmk/Task/
tem_dir/')
>label_dict_x=[label_dict[key] for key in
label_dict.keys()]
>label_pd = pd.concat(label_dict_x, axis=1,
sort=False)
>label_pd.index = cistopic_obj.cell_names
>label_pd.columns = ["pycisTopic_" + x for x in
label_pd.columns]
>cistopic_obj.add_cell_data(label_pd, split_
pattern = "-")
>import seaborn as sns
>fig, axs = plt.subplots(ncols = 3, nrows = 2,
figsize = (3 * 5, 2 * 5))
>for method, ax in zip(label_pd.columns.to_
list(), axs.ravel()):
    conf_mat = pd.crosstab(cistopic_obj.cell_
    data["Seurat_cell_type"], cistopic_obj.
    cell_data[method])
    conf_mat = conf_mat/ conf_mat.sum()
    sns.heatmap(conf_mat.loc[conf_mat.col-
umns], ax = ax)
```

Exporting to loom

```
>frompycisTopic.loomimportexport_region_accessibility_to_loom, export_gene_activity_to_loom
>cluster_markers = {"celltype": markers_dict}
>os.makedirs(os.path.join(out_dir, "loom"), exist_ok=True)
>export_region_accessibility_to_loom(
    accessibility_matrix = imputed_acc_obj,
    cistopic_obj = cistopic_obj,
    binarized_topic_region = region_bin_topics_otsu,
    binarized_cell_topic = binarized_cell_topic,
    selected_cells = cistopic_obj.projections["cell"]["UMAP"].index.tolist(),
    out_fname = os.path.join(out_dir, "loom," "10x_pbmc_pycisTopic_region_accessibility.loom"),
    cluster_annotation = ["celltype"],
    cluster_markers = cluster_markers,
    tree_structure = ("10x_pbmc," "pycisTopic," "noDBL_all"),
    title = "Tutorial - Region accessibility all,"
    nomenclature = "hg38,"
    split_pattern = "-")
>export_gene_activity_to_loom(
    gene_activity_matrix = gene_act,
    cistopic_obj = cistopic_obj,
    out_fname = os.path.join(out_dir, "loom," "10x_pbmc_pycisTopic_gene_activity.loom"),
    cluster_annotation = ["celltype"],
    cluster_markers = cluster_markers,
    tree_structure = ("10x_pbmc," "pycisTopic," "ATAC"),
    title = "Tutorial - Gene activity,"
    nomenclature = "hg38,"
    split_pattern = "-")
```

9.3 Preprocess scRNA-seq Data using Scanpy

```
>import scanpy as sc
>sc.settings.set_figure_params(dpi=80, frameon=False, figsize=(5, 5), facecolor="white")
>if not os.path.exists(os.path.join(work_dir, "scRNA")):
    os.makedirs(os.path.join(work_dir, "scRNA"))
>adata = sc.read_10x_h5(os.path.join(work_dir, "data/pbmc_granulocyte_sorted_3k_filtered_feature_bc_matrix.h5"))
>adata.var_names_make_unique()
>sc.pp.filter_cells(adata, min_genes=200)
>sc.pp.filter_genes(adata, min_cells=3)
>sc.external.pp.scrublet(adata)
>adata = adata[adata.obs["predicted_doublet"] == False]
>adata.var["mt"] = adata.var_names.str.startswith("MT-")
>sc.pp.calculate_qc_metrics(adata, qc_vars=["mt"], percent_top=None, log1p=False, inplace=True)
>import matplotlib.pyplot as plt
>mito_filter = 25
>n_counts_filter = 4300
>total_counts = adata.obs["total_counts"]
>pct_counts_mt = adata.obs["pct_counts_mt"]
>n_genes_by_counts = adata.obs["n_genes_by_counts"]
>adata = adata[adata.obs.n_genes_by_counts < n_counts_filter,:]
>adata = adata[adata.obs.pct_counts_mt < mito_filter,:]
>adata.raw = adata
>sc.pp.normalize_total(adata, target_sum=1e4)
>sc.pp.log1p(adata)
>sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3, min_disp=0.5)
>adata = adata[:, adata.var.highly_variable]
>sc.pp.scale(adata, max_value=10)
>adata_ref = sc.read("/share/home/zhangmk/Task/Scenicplus_example/pbmc_tutorial/data/pbmc3k_processed.h5ad")
>var_names = adata_ref.var_names.intersection(adata.var_names)
>adata_ref = adata_ref[:, var_names]
>adata = adata[:, var_names]
>sc.pp.pca(adata_ref)
>sc.pp.neighbors(adata_ref)
>sc.tl.umap(adata_ref)
>sc.tl.ingest(adata, adata_ref, obs='louvain')
>adata.obs.rename({'louvain': "ingest_celltype_label"}, inplace = True, axis = 1)
>sc.tl.pca(adata, svd_solver='arpack')
>sc.pp.neighbors(adata, n_neighbors=10, n_pcs=10)
>sc.tl.umap(adata)
>sc.pl.umap(adata, color = "ingest_celltype_label")
>sc.tl.leiden(adata, resolution = 0.8, key_added = "leiden_res_0.8")
>sc.pl.umap(adata, color = "leiden_res_0.8")
>tmp_df = adata.obs.groupby(["leiden_res_0.8," "ingest_celltype_label"]).size().unstack(fill_value=0)
>tmp_df = (tmp_df/ tmp_df.sum(0)).fillna(0)
>leiden_to_annotation = tmp_df.idxmax(1).to_dict()
> leiden_to_annotation["7"] = "B cells 1"
>leiden_to_annotation["11"] = "B cells 2"
>leiden_to_annotation = {cluster: leiden_to_annotation[cluster].replace(' ', "_") for cluster in leiden_to_annotation.keys()}
>leiden_to_annotation
>adata.obs["celltype"] = [leiden_to_annotation[cluster_id] for cluster_id in adata.obs["leiden_res_0.8"]]
>del(leiden_to_annotation)
>del(tmp_df)
> adata.write(os.path.join(work_dir, "scRNA/adata.h5ad"), compression='gzip')
```

9.4 Creating custom cistarget database

```
>git clone https://github.com/aertslab/create_cisTarget_databases
>wget https://resources.aertslab.org/cistarget/programs/cbust
>chmod a+x cbust
>mkdir -p aertslab_motif_colleciton
>wget -O aertslab_motif_colleciton/v10nr_clust_public.zip https://resources.aertslab.org/cistarget/motif_collections/v10nr_clust_public/v10nr_clust_public.zip
>cd aertslab_motif_colleciton; unzip -q v10nr_clust_public.zip
>REGION_BED="/home/Scenicplus_example/pbmc_tutorial/scATAC/consensus_peak_calling/consensus_regions.filtered.bed"
>GENOME_FASTA="/home/Scenicplus_example/pbmc_tutorial/data/hg38.fa"
>CHROMSIZES="/home/Scenicplus_example/pbmc_tutorial/data/hg38.chrom.sizes"
>SCRIPT_DIR="/home/scenicplus/ctx_db/create_cisTarget_databases"
>${SCRIPT_DIR}/create_fasta_with_padded_bg_from_bed.sh \
    ${GENOME_FASTA} \
    ${CHROMSIZES} \
    ${REGION_BED} \
    hg38.pbmc_with_1kb_bg_padding.fa \
    1000 \
    yes
```

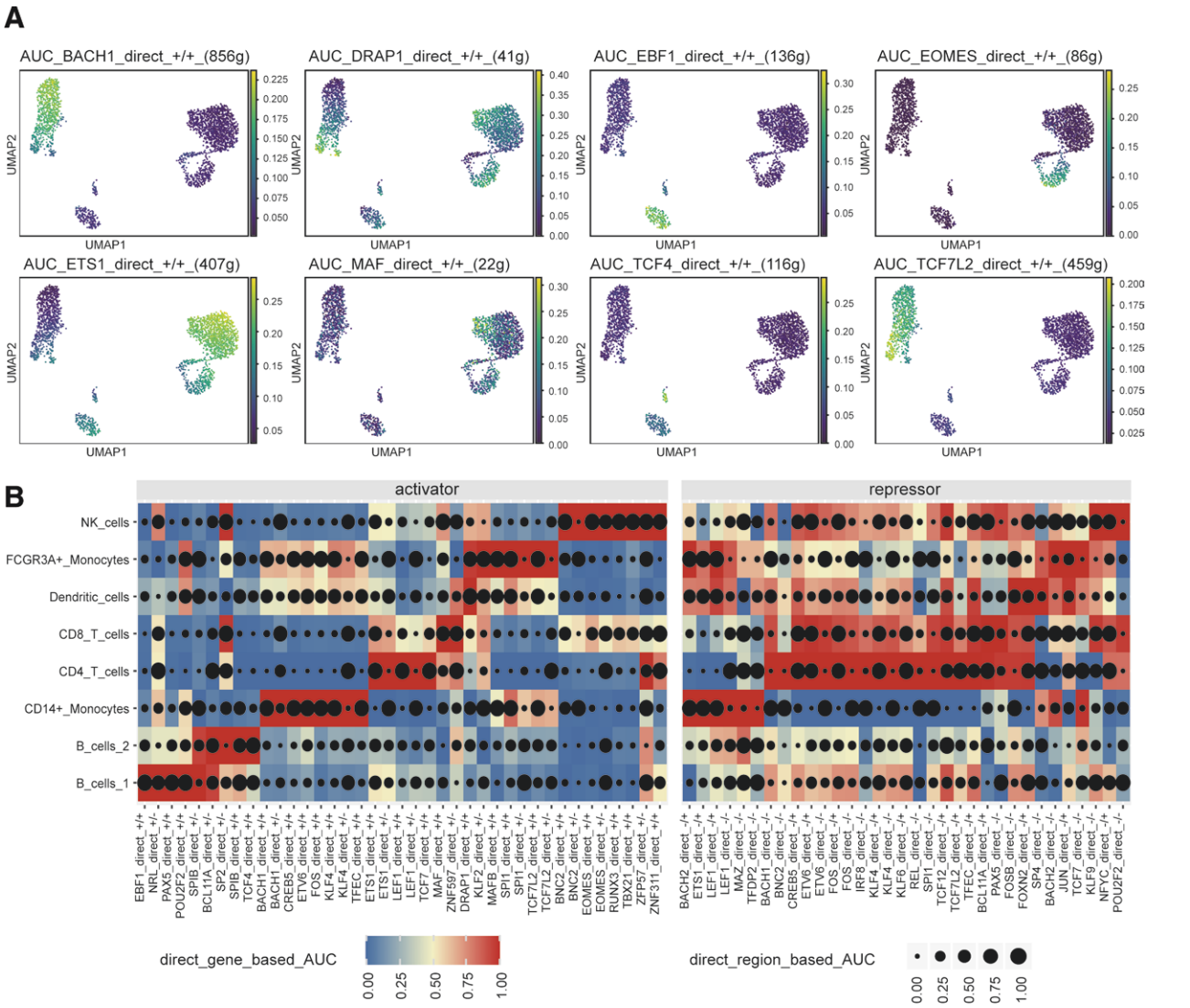

Figure 5. Multiscale visualization of selected eRegulon activity across single cells and immune cell types. (A) UMAP plots displaying gene-based AUC scores for manually selected direct eRegulons, representing key transcription factors (EBF1, TCF4, BACH1, ETS1, MAF, DRAP1, TCF7L2, EOMES). Each panel shows the single-cell-level activity of one eRegulon, with color intensity reflecting its regulatory activity (AUC). (B) Integrated dot-heatmap summarizing eRegulon activity across immune cell populations. The color scale encodes the gene-based AUC scores, while the dot size represents region-based AUC scores derived from chromatin accessibility. eRegulons are grouped into activators and repressors, and sorted by their gene-based activity. This figure illustrates the transcriptional and epigenomic regulation landscapes across immune subsets, combining both high-resolution single-cell views and aggregated cell-type patterns. AUC = area under the curve, UMAP = Uniform Manifold Approximation and Projection.

```
>ls aertslab_motif_colleciton/v10nr_clust_pub-
lic/singletons > motifs.txt
>OUT_DIR="
/home/Scenicplus_example/pbmc_tutorial/ctx_db"
>CBDIR="
${OUT_DIR}/aertslab_motif_colleciton/
v10nr_clust_public/singletons"
>FASTA_FILE="${OUT_DIR}/hg38.pbmc_with_1kb_bg_
padding.fa"
>MOTIF_LIST="
${OUT_DIR}/motifs.txt"
>DATABASE_PREFIX="10x_pbmc_1kb_bg_with_mask" "
>"${SCRIPT_DIR}/create_cistarget_motif_data-
bases.py"
    \-f "${FASTA_FILE}"
    \-M "${CBDIR}"
    \-m "${MOTIF_LIST}"
    \-o "${OUT_DIR}/${DATABASE_PREFIX}"
    \--bgpadding 1000 \
    -t 20
```

9.5 Configure Pipeline

Edit the config.yaml file with paths to your input files and analysis parameters:

```
>bat   /home/Scenicplus_example/pbmc_tutorial/
scplus_pipeline/Snakemake/config/config.yaml

# input_data:
  cisTopic_obj_fname:   "/home/Scenicplus_exam-
  ple/pbmc_tutorial/scATAC/cistopic_obj.pkl"
  GEX_anndata_fname:    "/home/Scenicplus_exam-
  ple/pbmc_tutorial/scRNA/adata.h5ad"
  region_set_folder: "/home/Scenicplus_example/
  pbmc_tutorial/scATAC/region_sets/"
  ctx_db_fname:      "/home/Scenicplus_example/
  pbmc_tutorial/ctx_db/10x_pbmc_1kb_bg_with_
  mask.regions_vs_motifs.rankings.feather"
  dem_db_fname:      "/home/Scenicplus_example/
  pbmc_tutorial/ctx_db/10x_pbmc_1kb_bg_with_
  mask.regions_vs_motifs.scores.feather"
```

```
path_to_motif_annotations:  "/home/Scenicplus_
example/pbmc_tutorial/ctx_db/motifs-v10nr_clust-
nr.hgnc-m0.001-o0.0.tbl"
```

9.6 Run the workflow using Snakemake

```
>snakemake --cores 24
```

9.7 Downstream analysis and visualization

After completing the workflow, analyze and visualize results:

To demonstrate the functionality of SCENIC+, we visualized the activity of selected eRegulons using UMAP projection and dot-heatmap representations (**Fig. 5A–B**). These results highlight the ability of SCENIC+ to integrate transcriptional and chromatin accessibility signals at both single-cell and population levels.

```
>import scanpy as sc
>import anndata
>import matplotlib.pyplot as plt
>eRegulon_gene_AUC = anndata.concat(
     [scplus_mdata["direct_gene_based_AUC"],
     scplus_mdata["extended_gene_based_AUC"]],
    axis = 1,)
>eRegulon_gene_AUC.obs    =    scplus_mdata.obs.
loc[eRegulon_gene_AUC.obs_names]
>sc.pp.neighbors(eRegulon_gene_AUC,  use_rep  =
"X")
>sc.tl.umap(eRegulon_gene_AUC)
>sc.pl.umap(eRegulon_gene_AUC,      color      =
"scRNA_counts:celltype")
>from scenicplus.RSS import (regulon_specificity_
scores, plot_rss)
>rss = regulon_specificity_scores(
    scplus_mudata = scplus_mdata,
    variable = "scRNA_counts:celltype,"
    modalities      =      ["direct_gene_based_
AUC,""extended_gene_based_AUC"])
>plot_rss(data_matrix = rss,top_n = 3,num_col-
umns = 4)
>sc.pl.umap(eRegulon_gene_AUC,      color      =
list(set([x for xs in [rss.loc[ct].sort_values()
[0:2].index for ct in rss.index] for x in xs})))
>from    scenicplus.plotting.dotplot    import
heatmap_dotplot
>from plotnine import ggplot
>p = heatmap_dotplot(
    scplus_mudata=scplus_mdata,
    color_modality="direct_gene_based_AUC,"
    size_modality="direct_region_based_AUC,"
    group_variable="scRNA_counts:celltype,"
    eRegulon_metadata_key="direct_e_regulon_
metadata,"
    color_feature_key="Gene_signature_name,"
    size_feature_key="Region_signature_name,"
    feature_name_key="eRegulon_name,"
    sort_data_by="direct_gene_based_AUC,"
    orientation="horizontal,"
    figsize=(16, 5))
```

Note on configuration: Based on our implementation experience, we highlight several practical issues for new users. First, specific versions of Polars (eg, 1.31.0) and the "pycistopic_v3" branch are required to avoid schema errors. Second, "pycisTopic," "pycistarget," and their dependencies (eg, pyarrow, anndata, numpy) may require manual installation prior to running SCENIC+. Finally, for offline environments, it is necessary to download genome annotations (eg, chromsizes) manually before executing Snakemake workflows. These points are included to improve reproducibility and reduce technical barriers.

Although our protocol was demonstrated on PBMCs from publicly available 10x Genomics data, the modular and generalizable nature of this workflow makes it suitable for future application to clinical samples. For example, single-cell chromatin accessibility and TF activity profiles could be leveraged to define regulatory subtypes of hematologic malignancies or to identify predictors of immunotherapy response in T cell–enriched samples.

However, applying scATAC-seq in clinical contexts still faces technical challenges. Low input cell numbers, partial degradation of clinical material (eg, frozen or formalin fixed paraffin embedded [FFPE] tissues), and sample heterogeneity can lead to sparse or noisy profiles, complicating downstream analysis. Emerging strategies such as targeted ATAC panels, advanced imputation models, and robust integrative pipelines may help overcome these limitations and bring single-cell epigenomics closer to translational use.

5. TROUBLESHOOTING

Problem 1: The output directory from Cell Ranger ATAC count is empty or missing essential files.

Potential Solution:

- Incorrect FASTQ file paths or filename misspellings;
- Incorrect or incomplete reference genome path;
- Insufficient computational resources causing the process to terminate prematurely.

Suggested solutions:

- Verify that all file paths are complete and correctly specified;
- Ensure the reference genome has been fully decompressed and the path is accurately provided;
- Allocate at least 64 GB RAM and 8 CPU cores for this step.

Problem 2: What if scATAC-pro fails due to missing modules or dependencies?

Potential solution:

This is often caused by improper installation or misconfiguration of dependencies.

Suggested solutions:

- Confirm that all required dependencies (eg, Python ≥3.8, R ≥4.0, Bowtie2) are installed;
- Use make configure followed by make install to properly set up the environment;
- Review configure_user.txt to ensure key parameters like PEAK_CALLER and CELL_CALLER are correctly defined.

Problem 3: The clustering results are suboptimal.

Potential solution:

Use the addClusters() function with a custom resolution parameter; Try multiple values to assess the impact on clustering granularity; Recalculate UMAP embeddings after adjusting the resolution to ensure consistent downstream visualization.

ACKNOWLEDGMENTS

This work was supported by the Chinese Academy of Medical Sciences (CAMS) Innovation Funds for Medical Sciences (2024-I2M-3-001).

AUTHOR CONTRIBUTIONS

C.C. conceived and supervised the study and wrote the manuscript. M.Z. analyzed and interpreted data. Both authors approved the final version of the manuscript.

REFERENCES

[1] Crawford GE, Holt IE, Whittle J, et al. Genome-wide mapping of DNase hypersensitive sites using massively parallel signature sequencing (MPSS). *Genome Res* 2006;16(1):123–131.

[2] Song L, Crawford GE. DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. *Cold Spring Harb Protoc* 2010;2010(2):pdb.prot5384.
[3] Hesselberth JR, Chen X, Zhang Z, et al. Global mapping of protein-DNA interactions in vivo by digital genomic footprinting. *Nat Methods* 2009;6(4):283–289.
[4] Giresi PG, Kim J, McDaniell RM, Iyer VR, Lieb JD. FAIRE (Formaldehyde-Assisted Isolation of Regulatory Elements) isolates active regulatory elements from human chromatin. *Genome Res* 2007;17(6):877–885.
[5] Schones DE, Cui K, Cuddapah S, et al. Dynamic regulation of nucleosome positioning in the human genome. *Cell* 2008;132(5):887–898.
[6] Barski A, Cuddapah S, Cui K, et al. High-resolution profiling of histone methylations in the human genome. *Cell* 2007;129(4):823–837.
[7] Buenrostro JD, Giresi PG, Zaba LC, Chang HY, Greenleaf WJ. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nat Methods* 2013;10(12):1213–1218.
[8] Kelly TK, Liu Y, Lay FD, Liang G, Berman BP, Jones PA. Genome-wide mapping of nucleosome positioning and DNA methylation within individual DNA molecules. *Genome Res* 2012;22(12):2497–2506.
[9] Buenrostro JD, Wu B, Chang HY, Greenleaf WJ. ATAC-seq: a method for assaying chromatin accessibility genome-wide. *Curr Protoc Mol Biol* 2015;109:21.29.1–21.29.9.
[10] Zhao F, Ma X, Yao B, Lu Q, Chen L. scaDA: a novel statistical method for differential analysis of single-cell chromatin accessibility sequencing data. *PLoS Comput Biol* 2024;20(8):e1011854.
[11] Granja JM, Corces MR, Pierce SE, et al. ArchR is a scalable software package for integrative single-cell chromatin accessibility analysis. *Nat Genet* 2021;53(3):403–411.
[12] Stuart T, Srivastava A, Madad S, Lareau CA, Satija R. Single-cell chromatin state analysis with Signac. *Nat Methods* 2021;18(11):1333–1341.
[13] Fang R, Preissl S, Li Y, et al. Comprehensive analysis of single cell ATAC-seq data with SnapATAC. *Nat Commun* 2021;12(1):1337.
[14] Bravo Gonzalez-Blas C, Minnoye L, Papasokrati D, et al. cisTopic: cis-regulatory topic modeling on single-cell ATAC-seq data. *Nat Methods* 2019;16(5):397–400.
[15] Wang C, Sun D, Huang X, et al. Integrative analyses of single-cell transcriptome and regulome using MAESTRO. *Genome Biol* 2020;21(1):198.
[16] Yu W, Uzun Y, Zhu Q, Chen C, Tan K. scATAC-pro: a comprehensive workbench for single-cell chromatin accessibility sequencing data. *Genome Biol* 2020;21(1):94.
[17] R Core Team (2023). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing.
[18] Zhang Y, Liu T, Meyer CA, et al. Model-based analysis of ChIP-Seq (MACS). *Genome Biol* 2008;9(9):R137.
[19] Wickham H. *ggplot2: Elegant Graphics For Data Analysis*. 2nd ed. New York: Springer-Verlag; 2016.
[20] Schep AN, Wu B, Buenrostro JD, Greenleaf WJ. chromVAR: inferring transcription-factor-associated accessibility from single-cell epigenomic data. *Nat Methods* 2017;14(10):975–978.
[21] Bravo Gonzalez-Blas C, De Winter S, Hulselmans G, et al. SCENIC+: single-cell multiomic inference of enhancers and gene regulatory networks. *Nat Methods* 2023;20(9):1355–1367.
[22] Python Core Team. *Python: A Dynamic, Open Source Programming Language*. Wilmington, DE: Python Software Foundation; 2023.
[23] Satpathy AT, Granja JM, Yost KE, et al. Massively parallel single-cell chromatin landscapes of human immune cell development and intratumoral T cell exhaustion. *Nat Biotechnol* 2019;37(8):925–936.
[24] Granja JM, Klemm S, McGinnis LM, et al. Single-cell multiomic analysis identifies regulatory programs in mixed-phenotype acute leukemia. *Nat Biotechnol* 2019;37(12):1458–1465.
[25] Welch JD, Kozareva V, Ferreira A, Vanderburg C, Martin C, Macosko EZ. Single-cell multi-omic integration compares and contrasts features of brain cell identity. *Cell* 2019;177(7):1873–1887.e17.
[26] Korsunsky I, Millard N, Fan J, et al. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat Methods* 2019;16(12):1289–1296.
[27] Stuart T, Butler A, Hoffman P, et al. Comprehensive integration of single-cell data. *Cell* 2019;177(7):1888–1902.e21.
[28] Polanski K, Young MD, Miao Z, Meyer KB, Teichmann SA, Park J-E. BBKNN: fast batch alignment of single cell transcriptomes. *Bioinformatics* 2020;36(3):964–965.
[29] Lopez R, Regier J, Cole MB, Jordan MI, Yosef N. Deep generative modeling for single-cell transcriptomics. *Nat Methods* 2018;15(12):1053–1058.
[30] Luecken MD, Buttner M, Chaichoompu K, et al. Benchmarking atlas-level data integration in single-cell genomics. *Nat Methods* 2022;19(1):41–50.