Genome Biology

RESEARCH

Open Access

Benchmarking computational methods for single-cell chromatin data analysis

Siyuan Luo[1,2], Pierre-Luc Germain[2,3,4], Mark D. Robinson[2,3*] and Ferdinand von Meyenn[1*]

*Correspondence: mark.robinson@mls.uzh.ch; ferdinand.vonmeyenn@hest.ethz.ch

[1] Laboratory of Nutrition and Metabolic Epigenetics, Department of Health Sciences and Technology, ETH Zurich, Zurich, Switzerland
[2] Department of Molecular Life Sciences, University of Zurich, Zurich, Switzerland
[3] SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland
[4] Institute for Neuroscience, Department of Health Sciences and Technology, ETH Zurich, Zurich, Switzerland

Abstract

Background: Single-cell chromatin accessibility assays, such as scATAC-seq, are increasingly employed in individual and joint multi-omic profiling of single cells. As the accumulation of scATAC-seq and multi-omics datasets continue, challenges in analyzing such sparse, noisy, and high-dimensional data become pressing. Specifically, one challenge relates to optimizing the processing of chromatin-level measurements and efficiently extracting information to discern cellular heterogeneity. This is of critical importance, since the identification of cell types is a fundamental step in current single-cell data analysis practices.

Results: We benchmark 8 feature engineering pipelines derived from 5 recent methods to assess their ability to discover and discriminate cell types. By using 10 metrics calculated at the cell embedding, shared nearest neighbor graph, or partition levels, we evaluate the performance of each method at different data processing stages. This comprehensive approach allows us to thoroughly understand the strengths and weaknesses of each method and the influence of parameter selection.

Conclusions: Our analysis provides guidelines for choosing analysis methods for different datasets. Overall, feature aggregation, SnapATAC, and SnapATAC2 outperform latent semantic indexing-based methods. For datasets with complex cell-type structures, SnapATAC and SnapATAC2 are preferred. With large datasets, SnapATAC2 and ArchR are most scalable.

Keywords: Benchmark, ScATAC-seq, Clustering, Feature engineering, Dimensional reduction

Background

Recent advances in single-cell sequencing technologies have enabled the profiling of genome-wide chromatin accessibility and histone modifications and allowed the exploration of epigenetic landscapes within complex tissues. However, the analysis of single-cell chromatin data is challenging due to two main reasons. Firstly, state-of-the-art technologies such as single-cell ATAC-seq (scATAC-seq) [1, 2] and single-cell CUT &Tag (scCUT &Tag) [3] are based on DNA tagmentation, which produces

sparse and noisy signals due to the low copy numbers and rare tagmentation events. It has been estimated that only 1–10% of accessible regions are detected per cell compared to corresponding bulk experiments [4]. Secondly, unlike in single-cell RNA-seq data, there are no fixed feature sets for chromatin data. Usually, a set of genomic regions (e.g., bins or peaks) is first determined, and then the tagmentation events are counted within each region. For large genomes such as human and mouse, this leads to very high-dimensional data that not only raises challenges on the time and memory efficiency of the processing pipelines but also hinders the statistical analysis. On the other hand, it is commonly assumed that single-cell data is sampled from a cellular state space that is of much lower intrinsic dimensionality than the observed data [5–7]. Therefore, it is necessary and important to learn a low-dimensional representation of the data before further analysis.

In the past few years, there have been many efforts on improving feature engineering and dimensional reduction methods for scATAC-seq data. One idea is to use approaches that are originally designed for sparse and high-dimensional data (e.g., Latent Semantic Indexing and Latent Dirichlet Allocation from the natural language processing field) and directly apply them to the cell-by-region count matrix. Several popular methods fall into this broad category, although the underlying algorithms differ. For example, Signac [8] uses Latent Semantic Indexing (LSI), which is a linear dimensional reduction method consisting of a normalization step (e.g., Term Frequency-Inverse Document Frequency, TF-IDF) and Singular Value Decomposition (SVD); ArchR [9] employs an iterative procedure of LSI, in order to refine the feature selection during each iteration; cisTopic leverages Latent Dirichlet Allocation (LDA), a topic modeling method, to discern thematic structures; SnapATAC [10] uses diffusion maps, and SnapATAC2 uses Laplacian eigenmaps, both of which are non-linear dimensional reduction methods that work by constructing a graph representation of the data and then utilizing the eigendecomposition of some form of graph matrix. Another group of approaches first uses domain knowledge to aggregate the genomic region set into a much smaller set of meta-features such as motif hits, k-mers, and genes and then applies dimensional reduction methods such as PCA on the cell-by-meta-feature matrix. For example, BROCKMAN [11] uses gapped k-mer frequency of the DNA sequence around insertion points, SCRAT [12] allows the usage of motifs, DNase I hypersensitive site clusters, genes, or gene sets as features, and Cicero [13] calculates gene activity scores. A third type of method uses neural network models, such as PeakVI [14], which uses a variational autoencoder, and scBasset [15], which uses a convolutional neural network. Other ideas include integrating DNA sequence information, such as in scBasset and CellSpace [16].

Despite a large amount of available methods, there is currently no consensus on the best usage of these methods for scATAC-seq data. Chen et al. [4] did a benchmark on 10 methods and showed that SnapATAC, cisTopic, and Cusanovich2018 [17] outperform other aggregation-based methods. Since then, many new methods have been proposed [8, 9, 14–16], and an updated benchmark is desirable. Although a subset of methods has been frequently benchmarked in papers of new methods using a few popular datasets, it is hard to find an agreement between these benchmarking efforts. Therefore, a comprehensive and neutral benchmark effort is desired [18] to give an unbiased perspective on how these methods perform on a large variety of datasets.

One way to evaluate the feature engineering and dimensional reduction methods is to combine them with unsupervised clustering with the aim to identify cell types or cell states, which is a fundamental step for many downstream analysis [19]. Previous benchmarking studies [4] for scATAC-seq data have focused on comparing the clustering outcomes at a single predefined resolution [20]. However, determining the true number of clusters in advance is not always feasible, and as dataset complexity increases, the choice of clustering resolution becomes dependent on user-defined parameters and biological questions [21]. Given that alterations in the number of clusters can have a substantial influence on many evaluation metrics [22, 23], such evaluations may not fully capture the scenarios encountered by users during dataset processing and interpretation.

To provide a comprehensive assessment of the methods under investigation, we conducted our evaluation across three distinct levels: cell embeddings, graph structure, and final partitions. We employed a set of ten metrics to evaluate performance at each of these levels. By considering multiple aspects of clustering quality, our evaluation approach aims to provide a more thorough understanding of the strengths and limitations of each method. Based on our results, we provide guidelines for choosing analysis methods for different data types. Meanwhile, our data and analysis also provide a comprehensive framework for benchmarking common single-cell chromatin data analysis steps.

Results

Benchmark design

To get a comprehensive understanding of the method performance, we used 6 published datasets of divergent sizes and sequencing protocols and from different tissues and species (Additional file 2: Table S1). In the absence of perfect ground truth, we included datasets with annotations from different information sources, including RNA modalities, genotypes, FACS-sorting labels, or tissue of origins. This ensures that our evaluation is not biased by specific assumptions of the ground truth. The coverage and signal-to-noise ratio (measured by transcription start site enrichment score, TSSE) also vary a lot across datasets, suggesting that our data collection represents a wide range of realistic test cases from different experimental protocols.

The benchmarking pipeline (Fig. 1) starts from quality control (QC) and preprocessing to get the fragment files in BED format. These files serve as the input for each method. Then feature engineering and dimensional reduction are performed, and a cell embedding matrix is generated. This particular stage is where the various methods are applied. Subsequently, each embedding matrix is loaded into a common clustering and evaluation pipeline to get the clustering results.

At the feature engineering and dimensional reduction stage, we benchmarked 5 methods in 8 configurations (Fig. 1). Signac was included as a representative of LSI-based methods, and it uses a dataset-specific peak set as the genomic regions. Two different ways of defining this peak set were tested: (1) aggregate all cells for peak calling, or (2) first do coarse cell clustering, then do peak calling per cluster and use the merged peak sets from all clusters. The iterative LSI in ArchR was also included and tested on either genomic bins or merged peaks. In addition, we assessed SnapATAC and its recently updated version, SnapATAC2. For SnapATAC2, when calculating the pairwise

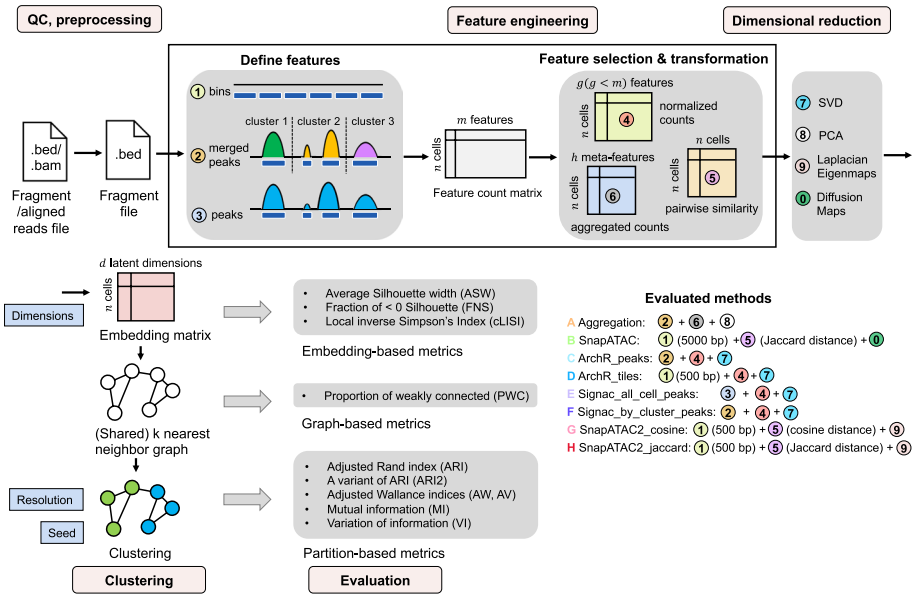

Fig. 1 Benchmark framework. Starting from the .bed format fragment files or the .bam format aligned reads files, barcode-level QC is performed, and the filtered fragment files are input to the benchmark pipeline. The feature engineering step consists of two stages: (i) defining the genomic features and (ii) feature selection and/or transformation. Approaches for each stage are listed, and respective approaches of each method are indicated. Next, dimensional reduction is performed to generate the cell embedding matrix. A shared nearest neighbor (SNN) graph is constructed from the embedding matrix and then used for Leiden clustering. Evaluations are conducted on the embedding matrix, SNN graph, and final partitions, each using different metrics. During the evaluation, we explored multiple values for parameters such as the number of latent dimensions, resolution, and random seed——their positions within the workflow are denoted by blue boxes

cellular similarity matrix, it allows the usage of either Jaccard or Cosine distance. So, we tested both metrics to see how appropriate they are for the sparse and near-binary data. Besides, while aggregating regions based on biologically meaningful features such as motifs has tended to have poor performance [4], ad hoc feature clustering and summing into meta-features was shown to be a viable strategy for doublet detection [24]. We therefore chose to include this strategy as well in our evaluation (for a more detailed description of each strategy, see the "Methods" section).

In the clustering process, several parameters could affect the clustering performance. We explored various values of these parameters to study their effects (Fig. 1). These parameters include the number of features (peaks or bins) been selected, the number of latent dimensions, resolution, and random seed used in Leiden clustering (see the "Methods" section). Meanwhile, during the evaluation process, we assessed each method at different clustering steps including the cell embedding, the shared nearest neighbor (SNN) graph, and the partition level, to eliminate the effect of potentially suboptimal parameter choice.

Method performance is dependent on the intrinsic structure of datasets

Among our six datasets, Cell line [9], Atlas1 [25], and Atlas2 [25] consist of mixed cell lines or cell types from various tissues. These three datasets show a relatively simple structure, with distinct cell clusters and little hierarchy. Conversely, the remaining

three datasets, derived from specific tissues, carry inherent complexity, including closely related subtypes and/or hierarchical structures. This division is reflected by the average levels of many evaluation metrics: the simpler datasets show a higher average Adjusted Rand Index (ARI) and lower cluster Local Inverse Simpson Index (cLISI) and Proportion of Weakly Connected (PWC) score and conversely (Figs. 2 and 3a). Through our analysis, we noticed that some methods performed relatively better on the simpler or more complex tasks.

As mentioned above, we first evaluated the cell embedding and SNN graph-level outputs (Fig. 2). The cLISI, which measures the purity of neighborhood composition in the embedding space, was always close to 1 in easy tasks and showed little discrimination between methods. This indicates that most local neighborhoods ($k = 90$) contain a single cell type in the embedding space. On the contrary, the Silhouette width is calculated between a cell and a whole cluster ($k > 300$). We observed that the average Silhouette width (ASW) sometimes showed inconsistent rankings compared to other metrics. While the Silhouette score has commonly been utilized for benchmarking clusterings in single-cell datasets, a potential issue is that Euclidean distance may not be suitable for accurate assessments in high-dimensional spaces over long-range distances [5]. Therefore, we consider the Silhouette score calculated using Euclidean distance at the cluster level to be less appropriate for our analysis. Nevertheless, we considered the fraction of

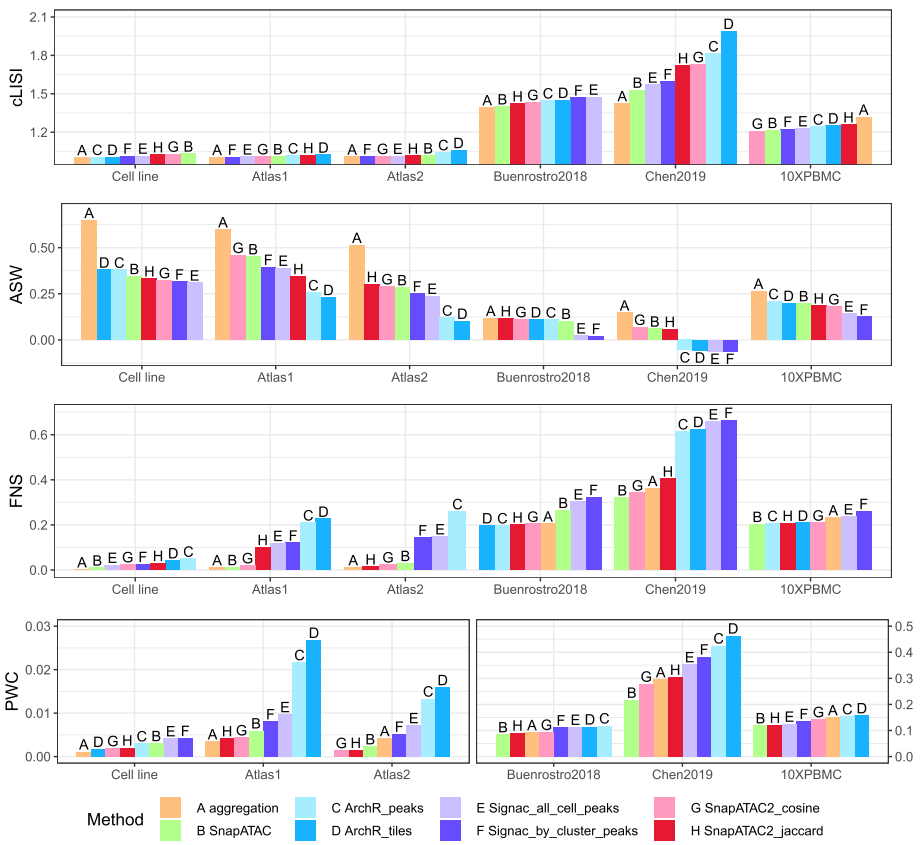

Fig. 2 Embedding- and graph-level metrics averaged across all cells or all cell classes for each dataset; each subpanel represents an evaluation metric. Bars are sorted from the best to the worst performance

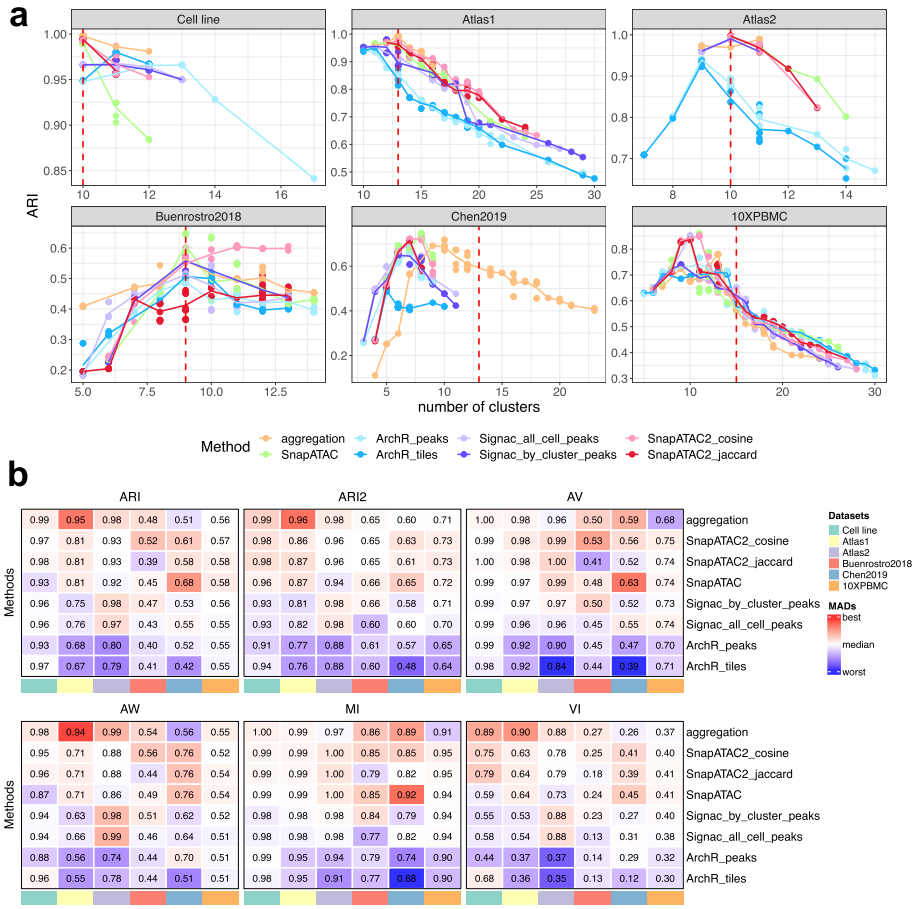

Fig. 3 Clustering results. **a** The adjusted Rand Index (ARI) plotted against various number of clusters; each subpanel represents a dataset. Each point represents a clustering solution obtained by varying the resolution parameter and the random seed in Leiden algorithm. The line plot is the average ARI at a given number of clusters. **b** Heatmaps displaying the normalized areas under the curve (AUC) from plots similar to **a**, but for various partition-level metrics. The color scale indicates deviations from the column-wise median scaled by the matrix-wise median absolute deviation [23]. It provides a comparison of relative performance between methods, and is unified across datasets and robust to outliers

negative Silhouette (FNS) to be relatively more robust to certain space transformations and thus more appropriate here. In two of the three relatively easy tasks, ArchR_peaks and ArchR_tiles exhibited the highest FNS, followed by Signac_all_cell_peaks and Signac_by_cluster_peaks, while SnapATAC, SnapATAC2_cosine, and aggregation always displayed close to 0 FNS. Consistently, ArchR showed the worst PWC score in two of the three easy tasks, followed by Signac.

At the clustering level, the 6 evaluation metrics measure multiple aspects of the clustering performance (Fig. 1). ARI, Variation of Information (VI) measure the overall agreement between the clustering results and the ground-truth annotation. The variant of ARI (denoted by ARI2) adjusts for the class size bias and is more sensitive to errors in small classes. One of the two Adjusted Wallance Indices, AV, and Mutual Information (MI) reflect mostly the homogeneity of clusters (i.e., the degree to which it includes only cells of one class), while the other Adjusted Wallance Indices, AW, represents the completeness of true classes in the clustering. Since the clustering solution varies by

using different resolutions and random seeds, these metrics would also vary across these parameters. As shown in Fig. 3a, ARI is strongly affected by the number of clusters. Usually, the best ARI is achieved at the cluster number that is equal or close to the ground-truth number of classes, and then as the cluster number increase or decrease, ARI can deteriorate dramatically. We therefore inspected multiple combinations of resolutions and random seeds that give different numbers of clusters and summarized the results using normalized areas under the curve (AUC, see the "Methods" section), as shown in Fig. 3b. The AUC of ARI showed that ArchR related methods tend to have a lower rank than SnapATAC2 based methods and that datasets 10XPBMC showed less discrimination between methods than other datasets, which is consistent with Fig. 3a. These observations confirm the use of AUC as a good summary of results across parameters.

For clustering tasks that are relatively easy, the number of clusters that provides the highest ARI value is usually equal to the number of classes of the ground truth (Fig. 3a), with a few exceptions. ArchR_tiles and ArchR_peaks achieved the best ARI using fewer clusters than the true classes in datasets Atlas1 and Atlas2, and then their performance starts to deteriorate as the number of clusters increases. This is because ArchR failed in separating small similar classes from one another and instead segregated other classes (Fig. 4a,b, Additional file 1: Fig. S1a-c,e). Signac_all_cell_peaks and Signac_by_cluster_peaks are among the best performing methods in the Atlas2 dataset (Fig. 3a, b) but also segregated classes in Atlas1 (Additional file 1: Fig. S1f,g). In contrast, SnapATAC, SnapATAC2_cosine, SnapATAC2_jaccard, and the aggregation method performed consistently well for the easy tasks in the sense that the optimal ARI is always achieved at the correct number of clusters (Fig. 3a); in addition, the clustering is nearly perfect (Fig. 4c,d). As noted in the cell line task, SnapATAC seems to have the worst performance according to ARI when over-clustering (Fig. 3a), but this is because it segregated a large class (293T) compared to what SnapATAC2 segregated (GM12878) (Additional file 1: Fig. S2b,c); this is an example of the cluster size bias of ARI. The adjusted version, ARI2, does not show a preference for SnapATAC2 over SnapATAC (Additional file 1: Fig. S2a), further underscoring the importance of considering multiple evaluation metrics. Other partition-based metrics indicate that aggregation, SnapATAC, and SnapATAC2 are the best methods, followed by Signac; ArchR performed worst (Fig. 3b). Overall, for easy tasks, SnapATAC, SnapATAC2, and the aggregation method performed the best , followed by Signac, while ArchR had a difficult time correctly clustering rare cell types.

(See figure on next page.)

Fig. 4 **a-f** True classes and their fractions of agreement with the predicted clusters. **a** and **b** are ArchR_tiles on Atlas1, **c** and **d** are aggregation and SnapATAC2_cosine on Atlas1, and **e** and **f** are aggregation and SnapATAC2_cosine on 10XPBMC. The *x*-axis is the predicted clusters, and the *y*-axis is the ground truth classes. The colors of tiles indicate the proportion of cells from the corresponding true class (each row sums up to one). A clearer diagonal structure indicates better agreement. ARI and ARI2 are calculated and shown on the top right. The bar plot on top shows the value of AV (the "Methods" section) and can be interpreted as the homogeneity of the corresponding clusters. The bar plot on the right shows the value of AW and represents the completeness of each true class in the prediction. The color of the bars shows the proportion of cells in each cluster/ground truth class. In title, the corresponding datasets, methods, and number of clusters are indicated. **g** Corresponding UMAP given by aggregation and SnapATAC2_cosine on dataset 10XPBMC. The aggregation method did not resolve correctly CD14 vs CD16 monocytes, as well as CD4 vs CD8 naive T cells

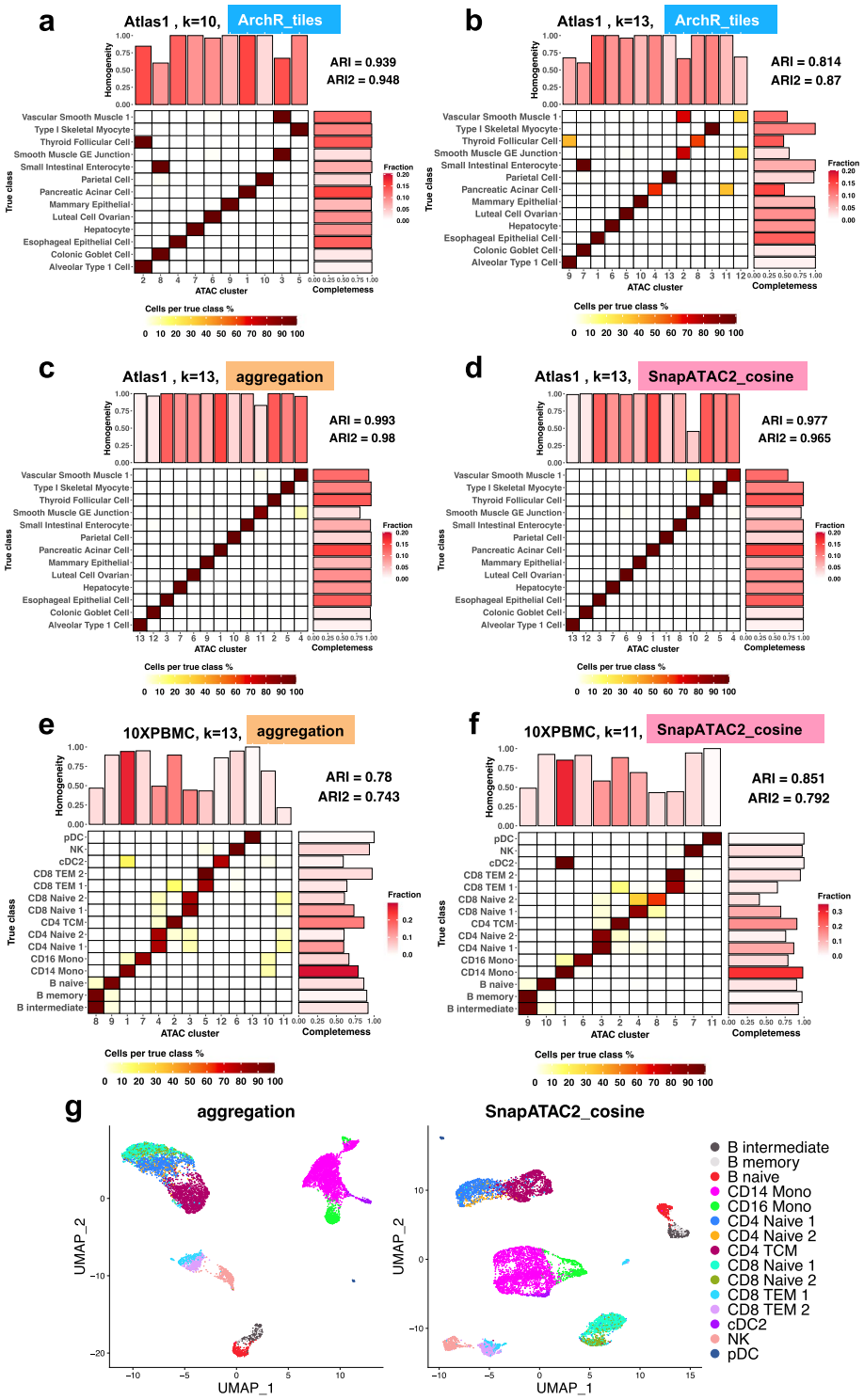

Fig. 4 (See legend on previous page.)

For complex datasets, the overall cLISI, FNS, and PWC increased compared to simple datasets (Fig. 2). Specifically, on average more than 20% cells of each true class have a negative Silhouette score, and more than 8% cells of each true class are weakly connected to the belonging communities in the SNN graph. In datasets Buenrostro2018 [26]

and 10XPBMC, cLISI and PWC did not show much discrimination between methods. FNS is also similar across methods in 10XPBMC, although slightly worse for Signac_all_cell_peaks, Signac_by_cluster_peaks, and SnapATAC in Buenrostro2018. In Chen2019 [27], the aggregation method exhibited the best local neighborhood purity reflected in the cLISI score, followed by SnapATAC, Signac_all_cell_peaks, and Signac_by_cluster_peaks, while ArchR_tiles and ArchR_peaks were the worst. FNS indicated that for Signac and ArchR, more than 60% cells of each true class have a negative Silhouette, while for other methods, this value is less than 40%. In alignment with this, ArchR and Signac also exhibited the worst PWC values in this dataset.

When comparing the clustering results, we noted that the number of clusters of the highest ARI does not always equal the number of classes in the annotation for difficult clustering tasks (Fig. 3a). This appears to be because of populations that are hard to separate, either because they are rare, as in the Chen2019 dataset (Additional file 1: Fig. S3), or because they are very similar to each other, as intermediate and memory B cells in the 10XPBMC dataset (Fig. 4e, f, g).

For these difficult clustering tasks, SnapATAC and SnapATAC2_cosine consistently performed the best, while ArchR_tiles and ArchR_peaks were the worst (Fig. 3a, b). SnapATAC2_jaccard seems to have a worse performance than SnapATAC and SnapATAC2_cosine in Buenrostro2018 according to ARI and AV, but it showed a comparative performance of ARI2. The aggregation method also showed good performance in Buenrostro2018 and Chen2019 but slightly underperformed in 10XPBMC. The decrease in performance of aggregation in 10XPBMC is mostly because of the mixing of subtypes, as AV and MI indicate (Fig. 3b), and is clear in Fig. 4e, g. Signac tended to perform better than ArchR no matter which configuration was used, and in dataset Buenrostro2018, it was comparable to SnapATAC and SnapATAC2. Overall, for difficult tasks, we found that SnapATAC, SnapATAC2_cosine, and SnapATAC2_jaccard are the top methods, followed by the aggregation approach, then Signac, while ArchR is the worst.

Method choices at different feature engineering steps

This section focuses on the analysis of various choices made during the process of feature engineering, specifically regarding genomic features, peak calling methods, and distance metrics, and their impact on the overall clustering performance.

Peaks versus bins

Due to the absence of a standard feature set for chromatin data, researchers often resort to using either genomic bins or peaks, each with its own limitations [28]. Genomic bins suffer from the arbitrary selection of bin length and break-up positions. On the other hand, peaks align more closely with functional intervals, but present challenges in their identification in rare cells, and require additional processing when integrating different datasets.

To assess the clustering performance with these two types of genomic features, we compared the results obtained using ArchR_tiles (non-overlapping genomic bins of 500bp) and ArchR_peaks (MACS-2 consensus peaks across clusters) (Additional file 1: Fig. S4a, Fig. S5a). Among most datasets we inspected, the performance of these two approaches is very similar across our metrics. This is consistent with claims in the

literature [29]. Only in the Chen2019 dataset, ArchR_peaks exhibited a slightly better cLISI score (Additional file 1: Fig. S4a) and higher clustering-level performance than ArchR_tiles (Additional file 1: Fig. S5a), mostly because it separated classes "L4 1" and "L4 2" better (Additional file 1: Fig. S3). In the Atlas2 dataset, the utilization of peaks demonstrated improved cluster homogeneity, as evidenced by metrics such as AV, MI, and FNS, but not necessarily enhanced class completeness in AW (Additional file 1: Fig. S4a, Additional file 1: Fig. S5a). In summary, we found that the use of peaks exhibited marginal or no significant improvement over the use of bins.

Peak calling methods

If peaks are used as the genomic features, to facilitate the identification of population-specific peak sets, a common approach is to employ a two-step peak calling procedure, in which cells are first clustered using global peaks, before a second round of per-cluster peak calls. In our study, we tested two Signac pipelines, namely Signac_all_cell_peaks and Signac_by_cluster_peaks, to compare the effectiveness of one-step versus two-step peak calling. We observed that at the embedding and graph level (Additional file 1: Fig. S4b), these two approaches showed nearly identical performance. At the final partition level, the performance was still very similar in easy tasks, and only on specific difficult datasets did one method perform slightly better than the other. Specifically, in the Buenrostro2018 dataset, Signac_by_cluster_peaks outperformed Signac_all_cell_peaks (Additional file 1: Fig. S5b), whereas in the Chen2019 dataset, Signac_all_cell_peaks yielded slightly better results (Additional file 1: Fig. S5b, Fig. S6a,b), mostly because L6 IT and L5/6 IT cells are not properly grouped by Signac_by_cluster_peaks. Overall, our findings revealed that the two-step approach does not always outperform the one-step approach.

Number of features

The methods benchmarked vary in their default settings, particularly in the default number of features they employ, which can differ by an order of magnitude. To investigate the contribution of this parameter to the different performances observed, we set ran the methods forcing them to use the same specific number of features (Additional file 1: Fig. S7). In general, we observed that ArchR appears to benefit from slightly higher number of features than used by default and that SnapATAC and SnapATAC2 did not perform well with such low number of features and required ideally 200k features. This highlights the importance of having methods capable of efficiently handling such high dimensionality.

Distance metrics

Despite the debate between using peaks or bins, scATAC-seq data is usually regarded as binary, and therefore in SnapATAC2, either Jaccard or cosine similarity was used to construct the affinity matrix. We observed in our results that both similarity metrics showed very similar cLISI and PWC scores across datasets; using cosine similarity gave better FNS scores in two of the six datasets (Additional file 1: Fig. S4c). Metrics of clustering results indicated that the performance of cosine similarity was very similar to Jaccard similarity (Additional file 1: Fig. S5c), especially after being adjusted for class size effects

using ARI2. Overall, our results proved that both similarity metrics work comparably well in these clustering tasks.

Dimensions of the latent space

The five methods we evaluated use different underlying algorithms for dimensional reduction. ArchR and Signac use truncated SVD, which identifies and preserves directions of maximum variance in the data. SnapATAC and SnapATAC2, on the other hand, apply graph-based spectral embedding. Specifically, SnapATAC2 uses Laplacian Eigenmaps, which removes higher-frequency variations from one node to its neighbors, and preserves low-frequency structures of the graph [30]. In the aggregation method, feature-level aggregation is performed, followed by principal component analysis (PCA). The goal is to exploit the redundancy of the high-dimensional genomic feature space to average out potential noise.

Considering the varying assumptions and goals of each dimension reduction method, we investigated if different numbers of dimensions across methods could contain distinct information and how the choice of the number of dimensions for the embedding space affects performance. We examined a series of d values, namely 15, 30, 50, 100, and calculated the embedding-level and graph-level evaluation metrics (see Additional file 1: Fig. S8a). We observed that SnapATAC and SnapATAC2 were particularly sensitive to this parameter, with performance rapidly deteriorating as d increased. This trend is observable in the increasingly blurred structures in UMAP in Additional file 1: Fig. S8b and suggests that the later dimensions may contain less cell-type-relevant information. In contrast, the aggregation method demonstrated robustness to this parameter across most datasets. This aligns with the assumption that the aggregation method removed the noise by averaging it out, so that later dimensions are also smoothed signals. Signac and ArchR displayed an intermediate trend, and later eigenvectors may also have a smaller signal-to-noise ratio, especially in more complex datasets.

Stability of clusterings and robustness of clustering performance

We observed that in certain cases, the inherent randomness of Leiden algorithm (the "Methods" section) can lead to instability in the clustering results (e.g., Additional file 1: Fig. S9). To account for this, we performed the clustering steps using 5 different random seeds and compared the clustering results by calculating pairwise ARI (Additional file 1: Fig. S10 and Fig. S11). The variability in the clustering outcomes varies depending on the datasets, methods, and resolution parameters employed. Generally, using the same resolution value yielded partitions with the same number of clusters. However, in some cases, changing the random seed resulted in partitions with different cluster numbers, leading to increased variability. Notably, in simple datasets, forcing over-clustering of cells by increasing resolution tends to amplify the variability (Additional file 1: Fig. S11). This can be attributed to the fact that the simpler datasets only have flat clustering structures, and increasing the resolution merely introduces random splits within the true communities.

We then focused on the resolution value that yielded the highest clustering performance, as measured by ARI against the ground truth. Specifically, we looked at how much the pairwise ARIs between seeds deviate from 1, which reflects the level of

instability in the clustering outcomes. Interestingly, we found a positive correlation between the deviation and PWC value of the SNN graph (Additional file 1: Fig. S12a). Furthermore, we observed that the variation in clustering outcomes was more prominent across datasets rather than between methods.

In order to assess the impact of clustering result variability on the evaluation of clustering performance, we examined the variability of performance measurements using different random seeds. Notably, despite the instability observed in the clustering results, we found that the similarity to the ground truth remained relatively consistent (Fig. 5a, coefficient of variation (CV) of ARI; Additional file 1: Fig. S13). This consistency provides us with a solid foundation for confidently interpreting the evaluation results.

LSI-based methods show strong library size biases

Large library size variation arising from technical biases are often observed in single-cell data, and can potentially confound the downstream analysis [19]. Therefore, we examined to what extent the cell embedding of each method was driven by library size. By looking at the scatter plot of each latent dimension against the empirical library size, one can see that LSI-based methods (Signac, ArchR) showed a strong library size bias across all datasets (Additional file 1: Fig. S14). The difference between cell types may reflect biological variation, since global chromatin accessibility can differ during cell differentiation. However, the difference within each cell type is more likely due to technical aspects such as sampling effects. Therefore, we quantified this bias by calculating, for each latent dimension, the Pearson's correlation coefficient r with the square root of library size per cell type, and averaged the absolute value across all cell types. This value is further averaged across the first 5 components (Fig. 5b, average absolute correlation). Note that we followed the suggestion in Signac's tutorial and always removed latent components $r > 0.75$ with the library counts. In all our datasets, this criterion always removed the first component of Signac and ArchR embeddings, while no component was removed

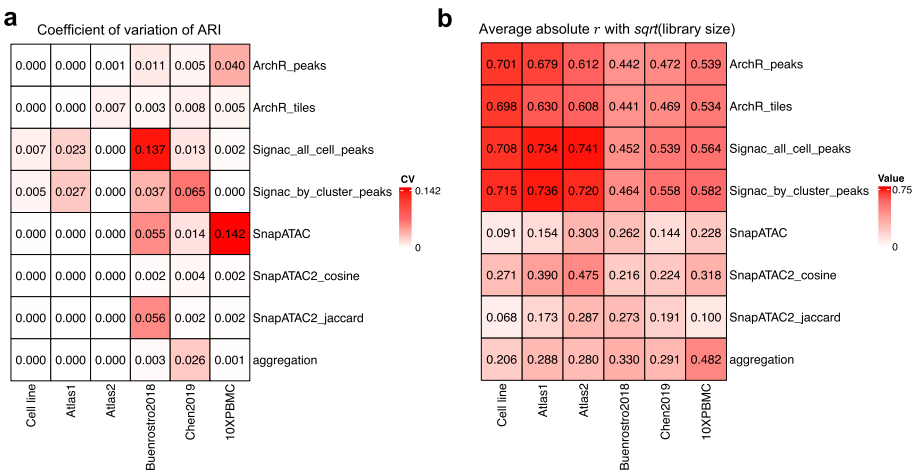

Fig. 5 **a** Coefficient of variation of ARI between predicted clusterings and the true cell types. **b** Average absolute Pearson's correlation with fragment counts; for a given latent dimension and a given cell type, the Pearson's correlation coefficient is calculated between the latent axis and the square root of fragment counts. This correlation value was then averaged across cell types and across the first 5 latent components

for other methods. After this filtering, the average absolute correlation is still between 0.5 to 0.75 in Signac and ArchR, followed by aggregation-based method and SnapATAC2 using cosine distance, where the correlation is around 0.2–0.5, then SnapATAC and SnapATAC2 using Jaccard distance, with correlation smaller than 0.3. We further evaluated how the library size bias affects the whole cell embedding by calculating the spatial autocorrelation of library size on the k nearest neighbor graph using Geary's C index (Additional file 1: Fig. S12b). Signac and ArchR tend to have a positive spatial autocorrelation, while aggregation, SnapATAC, and SnapATAC2 tend to have a smaller autocorrelation. In conclusion, LSI-based methods such as Signac and ArchR generate latent representations that are strongly associated with library size, while SnapATAC and SnapATAC2 using Jaccard distance are less affected.

Benchmarking methods for predicting gene activity scores

Besides clustering cell types, another typical analysis of scATAC-seq data involves predicting gene activities. In the five methods we discussed, four offer capabilities for inferring gene activities. ArchR's [9] study included a comparative analysis of various gene score models and found that ArchR's optimal model enhanced gene score prediction over other methods. Here, we independently benchmarked these four methods for gene activity prediction, presenting our results in a manner akin to ArchR's study for comparative purposes. By using the two multi-omics datasets, we compared the correlation between the RNA data and the predicted gene scores. Despite the prevalence of zeros and ties in gene expression and gene activity data due to high dropout rates, we found that the outcome rankings between methods using Pearson's correlation (Fig. 6) closely matched those obtained with Kendall's correlation (Additional file 1: Fig. S15). By looking at per-gene correlations across cells and metacells (as shown in Fig. 6a, b, e, and f), we observed that ArchR slightly outperformed Signac in the 10XPBMC dataset, yet it was less effective in the Chen2019 dataset. Contrary to the findings reported in ArchR's study, SnapATAC and SnapATAC2 yielded better results than both ArchR and Signac in several instances (refer to Fig. 6a, c–h). These findings underscore the importance of conducting neutral and independent benchmarking of computational tools.

Differences in scATAC-seq processing propagate to integration with scRNA-seq

We also evaluated the impact of different scATAC-seq processing techniques on their ability to integrate with scRNA-seq data. We performed cross-modality integration using the two multiome datasets by treating the cells from two modalities as if they were unmatched (i.e., unpaired integration) and used pairing information as the ground truth for evaluation. An effective ATAC processing method should ideally bring the corresponding matched cells into close proximity within the integrated space. We measured the fraction of samples closer than the true match (FOSCTTM) [31] after integration (see the "Methods" section). Although performances of the various methods were comparable (Additional file 1: Fig. S16), the differences were largely consistent with our evaluations on previous tasks (e.g., SnapATAC/SnapATAC2 generally providing better integration, and ArchR underperforming).

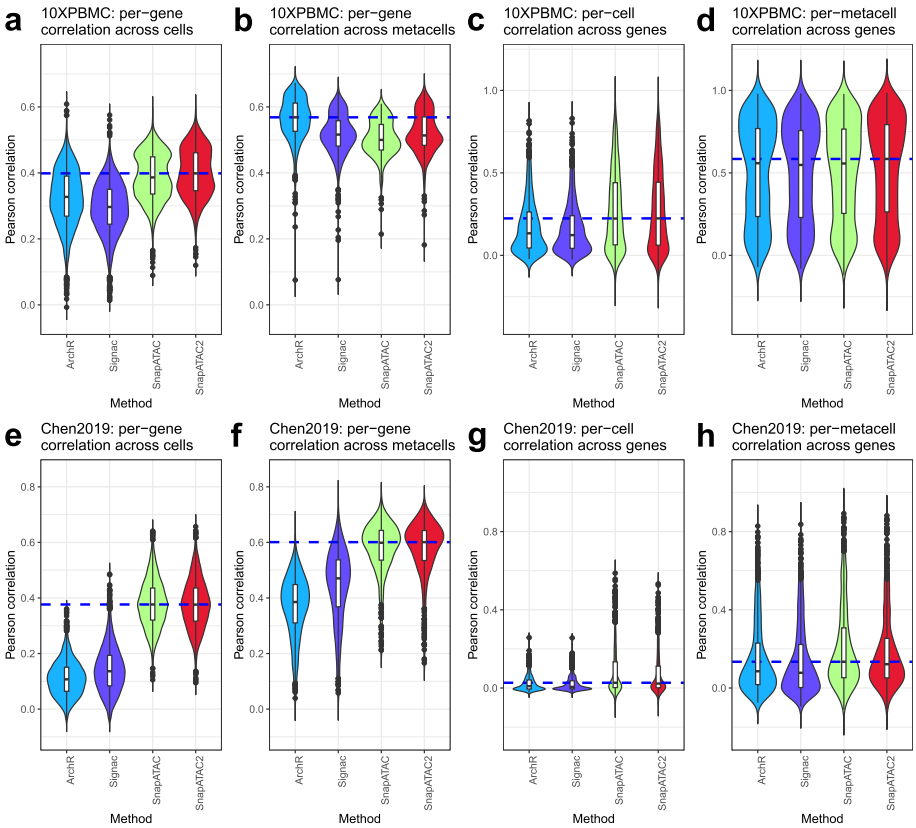

Fig. 6 Distribution of Pearson's correlations between the inferred gene activity score and the aligned gene expression. **a–d** are data from 10XPBMC dataset, and **e–h** are data from Chen2019 dataset. In **a**, **b**, **e**, and **f**, the per-gene correlations are calculated across cells (**a**, **e**) or metacells (**b**, **f**) (500 metacells in total). In **c** and **g**, the per-cell correlations are calculated for each gene. In **d** and **h**, the per-metacell correlations are calculated for each gene. The blue dashed line represents the median value of the best-performing model. Violin plots represent the smoothed density of the distribution of the data

Time and memory complexity

Due to the large feature space in scATAC-seq data, it is crucial to use methods that scale efficiently in terms of time and memory usage. We monitored the CPU time and peak memory usage in our Snakemake pipeline (Fig. 7a, b). For the aggregation method, we tracked the program either from the start of peak count matrix generation (aggregation + Signac) or subsequent to it. We found that SnapATAC2 performed the best in terms of runtime, while ArchR was the most memory efficient. SnapATAC had low memory consumption with small datasets; however, its memory usage increased rapidly as the dataset size increase, making it the least scalable option.

ArchR and SnapATAC2 both use on-disk storage instead of loading the entire dataset into memory. This is achieved by storing large-sized data in an HDF5-format file on disk and using an object to store small-sized metadata, which contains references to the corresponding files on disk and facilitates synchronization between the on-disk and in-memory data representations. This strategy makes them memory efficient and particularly well-suited for handling scATAC-seq data. For example, it enables ArchR and SnapATAC2 to handle objects that use genome-wide bins with a size as small as 500bp. In our analysis on number of features, ArchR had a increasing peak memory usage as the

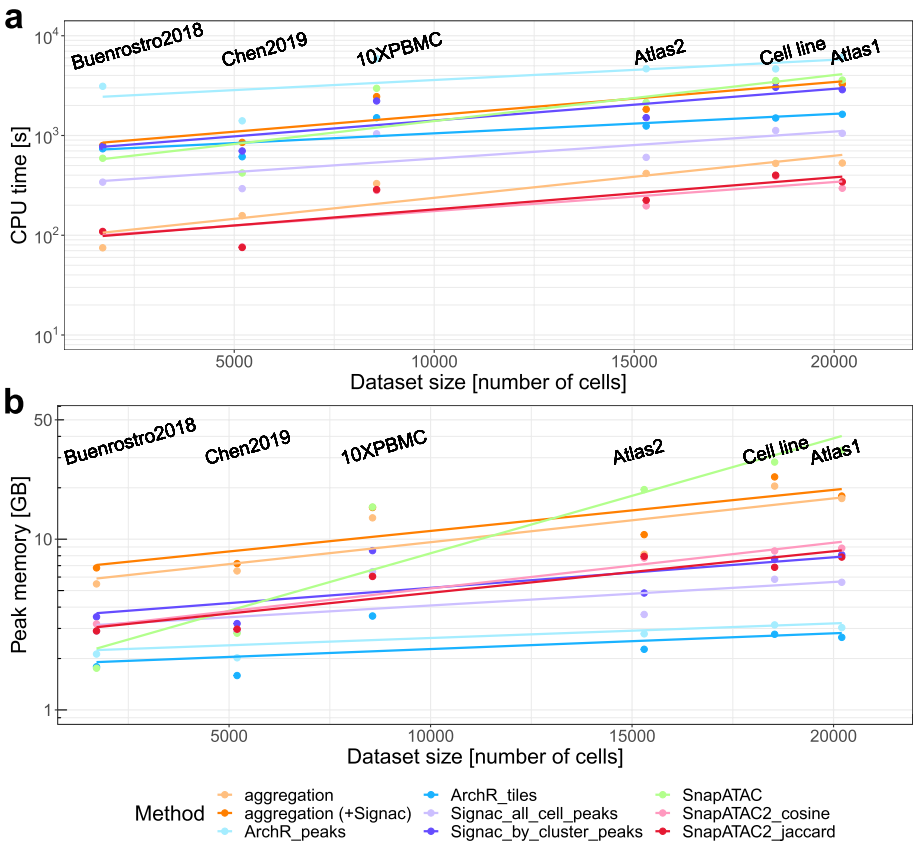

Fig. 7 The CPU time and peak memory usage of each method across datasets of different sizes

feature number increases (Additional file 1: Fig. S17), although the impact of the feature count on the memory and computational time was less pronounced than the effect of the cell count (Additional file 1: Fig. S17 and Fig. S18).

Depending on the method for feature matrix construction, the running time for aggregation can vary. The aggregation steps are relatively fast, e.g., within 10 min for 20000 cells. For ArchR and Signac, it depends on how the genomic features are defined. Unsurprisingly, ArchR_peaks nearly doubled the running time compared to ArchR_tiles, and Signac_by_cluster_peaks doubled the time compared to Signac_all_cell_peaks. This is because ArchR_peaks/Signac_by_cluster_peaks performed a second round of processing on top of ArchR_tiles/Signac_all_cell_peaks. ArchR_peaks and Signac_by_cluster_peaks aim to trade speed for improved identification of small cell classes.

Discussion

We benchmarked 8 data processing pipelines derived from 5 different methods developed for scATAC-seq data, focusing on their capability to discern cell heterogeneity and delineate cell types. By using 10 metrics to assess the performance at the embedding, graph, and partition levels, we systematically examined each pipeline and evaluated the impact of key parameter choices at each data processing stage. We observed that the ranking of methods is dependent on the complexity of the datasets. For simpler datasets with distinct cell types, aggregation outperformed other methods and demonstrated

superior performance in identifying small cell classes. SnapATAC2 emerged as the second-best method, while ArchR and Signac struggled to identify rare types. For complex datasets presenting hierarchical clustering structures and highly similar subtypes, SnapATAC and SnapATAC2 proved to be the most efficient in distinguishing subtypes. The aggregation method is second best, although occasionally it failed to detect differences between subtypes. Finally, SnapATAC and SnapATAC2 were also the best overall methods to infer gene activity scores.

Our evaluation metrics measured the efficacy of feature engineering at each stage of the clustering pipeline, including cell embedding learning, SNN construction, and graph-based partitioning. On one hand, this approach allows us to dissect distinct facets of clustering performance. For instance, ARI2 offers a sensitive measure of the quality of rare cell type identification, while PWC, at the SNN graph level, quantifies the extent of isolation between cell types and is well-suited for evaluating both small and closely related classes. On the other hand, our metrics facilitate a rigorous evaluation that is not confounded by potentially suboptimal parameter choices at intermediate stages. For example, the AUC of ARI provides an overall performance summary across different resolutions, while cLISI assesses the cell embedding, which is at the stage prior to the determination of resolution and partitioning. The ranking of methods defined by different metrics is not always consistent with each other, which is a common observation in various benchmarking efforts. This further highlights the importance to incorporate multiple metrics and allows users to focus on the most relevant aspects of the evaluation according to their biological questions.

We built our benchmark on 6 datasets with different types of annotations serving as the ground truth. Among these, 4 datasets contain annotations from either genotype, tissue origins, or FACS labels, which we regard as high confidence for those specific datasets. The remaining 2 datasets are multi-omic data, where RNA modality is used to infer annotation. In these two datasets, we observed that the number of natural clusters does not always agree between RNA and ATAC, and the best ARI is not always achieved at the number of clusters of RNA data. While there may be discrepancies between RNA and ATAC classes, for example if epigenetic differences lack a transcriptomic correlate, these discrepancies are unlikely to significantly bias our comparisons. However, for cell state differences that are specific to epigenetic changes but lack transcriptomic alterations, our benchmark will not be able to include their comparison. In such cases, multi-omics datasets with multiple epigenomic layers might help.

A potential limitation of our study lies in the composition of our datasets, which predominantly consist of well-defined cell types rather than a spectrum of continuous cell states. This could bias our evaluation in favor of methods that facilitate clear separation between distinct states. In scenarios where mapping a continuous trajectory is critical for downstream analyses, the preferred method might differ. However, cell-type clustering remains at the heart of single-cell data analysis. Identifying the most effective feature engineering method to discern cell type differences also contributes towards establishing a foundation for studying state-informative features in future studies.

We believe that our benchmark not only provides practical guidance for users in choosing methods for their biological analyses but also illuminates areas for potential improvements in future method development. First, while previous benchmarks have

concluded that methods based on aggregating accessible chromatin regions at the motif or gene level generally underperform [4], our benchmark illustrates that a purely data-driven aggregation strategy can achieve top performance. This suggests that redundant information in scATAC-seq data exists and can be harnessed to reduce noise.

Our analysis also highlights the challenge of mitigating library size effects. Library size effects, caused by technical variations, have long been observed in next-generation sequencing data, and normalization steps to correct these are now standard practice for single-cell RNA-seq data [19, 32, 33]. However, in the context of scATAC-seq data, this issue has not been adequately characterized or addressed. One aim of TF-IDF transformation performed in Signac is to correct for the library size difference between cells. From our observation, this is not very efficient. Linear regression-based normalization implemented in SnapATAC and SnapATAC2 seems to work well, but further comparison is needed. The binarization of peaks or bins can also be regarded as a normalization strategy [33], but recent work has also shown that retaining the count information instead of binarizing it can improve the performance of some models [28, 34], indicating that even in single cells, chromatin accessibility may actually be quantitative. In summary, striking a balance between removing the technical variance and avoiding excessive correction that could mask biologically meaningful differences in global or local accessibility levels remains a challenge.

The field of computational methods for scATAC-seq data is continually advancing, with new methodologies regularly emerging. There is an ongoing need for robust and neutral benchmarking efforts that serve both method users and developers effectively. While we have incorporated the most prevalent and recent methods in this study, we acknowledge that the immediacy of our work will inevitably diminish over time. To facilitate future benchmarking work, we offer a reproducible and expandable Snakemake pipeline of our benchmarking framework, and have made our processed datasets and intermediate data publicly available.

Conclusions

Taking together, we suggest choosing method for scATAC-seq analysis according to the complexity and size of the targeted dataset. For datasets with a simple structure where cell types are distinct from each other, all methods generally perform well; if small cell classes are expected and of interest, the aggregation method, SnapATAC, or SnapATAC2 are preferred. For more difficult tasks with hierarchical clustering structures and highly similar subtypes, SnapATAC and SnapATAC2 are among the best choice.

When the dataset is large (e.g., more than 20000 cells), SnapATAC is not very memory efficient on a typical desktop computer, and SnapATAC2 is preferred. Signac generally performs better than ArchR, but ArchR is more memory efficient. Aggregation steps do not add much time and memory consumption on top of Signac, so whenever Signac is used, the aggregation method can also be performed easily.

During the feature engineering steps, our results suggested that the choice peaks versus bins, or one-step versus two-step peak calling, are usually comparable in their performance. Users can choose according to their preferences. If SnapATAC or SnapATAC2 is used, a dimension of the latent space between 10 and 30 is recommended. For Signac

and ArchR, 10 to 50 dimensions represent a reasonable range, while for aggregation, a larger number of dimensions is still suitable.

Methods

Datasets and preprocessing

For our benchmark, we used 6 scATAC-seq or single-cell multi-omics datasets that are publicly available [9, 25–27] (see Additional file 2: Table S1; links to the public repositories can be found in the "Availability of data and materials" section). For datasets where the fragment files are publicly available, these were downloaded from the author's repositories. For datasets where the fragment files are not available, we downloaded the bam files and used the command line tool Sinto [35] to create fragment files.

For each scATAC-seq (or the ATAC component of the multi-omic) dataset, we first performed per-cell quality control (QC) using ArchR (v1.0.3) [9] by thresholding the Transcription Start Site Enrichment Score (TSSE) and the number of unique fragments; the thresholds for each dataset are in Additional file 2: Table S2. Then, we applied doublet removal procedures using `addDoubletScores()` and `filterDoublets()` in ArchR. Key parameters for these two functions are in Additional file 2: Table S2, including k in function `addDoubletScores()` and filterRatio in function `filterDoublets()`. We then filtered the fragment files to keep only cells that passed QC. For single-cell multi-omics datasets, we filtered by QC of both the ATAC and the RNA modalities. QC of RNA-seq was conducted using Seurat (v4.3.0) [36] by applying filters `nCount_RNA>800 & percent.mt< 5` for Chen2019 and `nFeature_RNA> 200 & nFeature_RNA< 5000 & nCount_RNA< 25000 & percent.mt< 20` for 10XPBMC. Doublets were identified using function `scDblFinder()` in R package scDblFinder (v1.13.9) [24] and then removed. Before calling `scDblFinder()`, Louvain clustering [37] was performed with resolution= 0.5 for Chen2019 and 0.8 for 10XPBMC in Seurat. Then, the clusterings results were used in `scDblFinder()`. All these filtering steps were applied to the fragment files, and the final filtered fragment files were inputs for the Snakemake pipeline.

Datasets from the human adult single-cell chromatin accessibility atlas

The human adult single-cell chromatin accessibility atlas [25] contains 111 distinct cell types across 30 tissues and is a rich resource for scATAC-seq data of different cell types. We took two subsets of this atlas as our evaluation datasets "Atlas1" and "Atlas2." The idea is to use the tissue of origin as the ground truth for benchmarking clustering. By examining the fraction of cells from different tissues for each cell type, we found that many cell types exist exclusively in one tissue (Additional file 1: Fig. S19). Therefore, we selected tissue-cell-type pairs by first selecting a subset of cell types that have $\geq$ 85% of cells from the same tissue, and then for each of the corresponding tissues, we selected one cell type randomly but excluded cell types that have less than 300 cells. For each tissue with multiple samples, we selected one sample that contains the maximum number of cells of that cell type that have passed QC. We used only one sample for each cell type to eliminate any potential batch effect. For the tissue-cell-type pairs selected for each dataset, we downsampled cells per cell type, using fractions 0.3 for Atlas1 and 0.5 for

Atlas2. The tissue-cell-type pairs and the sampled cell ID for "Atlas1" and "Atlas2" are available on GitHub [38].

scRNA-seq data annotation

For 10XPBMC and Chen2019 datasets, the RNA modality was subjected to Leiden clustering in Seurat using resolution = 0.5 for Chen2019 and resolution = 0.8 for 10XPBMC. The resolution for each dataset was determined by first performing Leiden clustering using a series of resolution values (with the maximum resolution obviously over-clustering the datasets), then constructing a cluster tree showing the co-clustering consistency across resolutions [39], and finally choosing a resolution value that gives stable clustering result and reasonable separation of cells in the UMAP. Then, for each cell, a label was transferred by reference mapping [36] using Seurat. In the case of the 10XPBMC dataset, an annotated PBMC reference dataset [40] was utilized for label transfer. As for the Chen2019 dataset, the scRNA-seq data of the adult mouse brain from the Allen Brain Atlas [41] served as the reference dataset. Then, we performed some manual curation to get the final cluster annotation. We describe roughly this process below. For each Leiden cluster, the majority cell label was token as the label of that cluster. If two clusters got the same label, we subset all cells from these two clusters and performed multiple rounds of clusterings on this subset. If the splitting of these two clusters were stable, we labeled them differently, including "CD4 Naive 1" and "CD4 Naive 2." Otherwise, we merged these two clusters.

Feature engineering methods

For all methods, we followed the procedures recommended in the author's documentation.

Signac

Starting from the fragment file, Signac first uses MACS2 for peak calling, then performs LSI on the peak count matrix to obtain a low-dimensional representation. Peak calling was conducted in two ways: (1) aggregating all cells for peak calling (denoted as "Signac_all_cell_peaks") or (2) aggregating cells and calling peaks for each cluster individually, followed by generating a consensus peak set from the peaks identified in all clusters, referred to as "Signac_by_cluster_peaks." LSI consists of 3 steps: (1) normalization using term frequency-inverse document frequency (TF-IDF), (2) selecting the top 95% most common peaks, (3) performing singular value decomposition (SVD).

We used the R package Signac (v1.9.0) for its implementation. As suggested by the tutorial: https://stuartlab.org/signac/articles/pbmc_multiomic.html, we created a fragment object, called MACS2, and removed peaks on nonstandard chromosomes and genomic blacklist regions and peaks having width < 20 or width > 10000. To identify peaks per cluster, the cell embeddings generated by "Signac_all_cell_peaks" are used to define clusters using the Louvain algorithm with a default resolution of 0.8.

Subsequently, we performed normalization, feature selection, and linear dimensional reduction using `RunTFIDF()` with `method=1`, `FindTopFeatures()` with `min.cutoff="q5"`, and `RunSVD()` with `n=100`, respectively. As suggested in the tutorial,

the first LSI components often capture sequencing depth. We removed LSI components that have larger than 0.75 Pearson correlation with the total number of counts.

Aggregation

The aggregation method starts with the peak count matrix where the peak set is identified using the method "Signac_by_cluster_peaks." Then, the cell-by-peak fragment count matrix is used for subsequent TF-IDF normalization and PCA. Minibatch K-means clustering is applied to the PCA to cluster peaks into meta-features ($K = 1000$ by default). Ultimately, an aggregated count matrix is obtained by summing the counts per meta-feature, and PCA is performed on the aggregated count matrix to get the low-dimensional representation.

We used the function `aggregateFeatures()` in R package scDblFinder (v1.13.9) with the default parameters. By default, $K = 1000$ feature clusters are identified.

ArchR

ArchR takes the fragment files as input and can use either the genomic tiles or peaks as features. We implemented both options. The "ArchR_tiles" method uses 500-bp non-overlapping genomic tiles to construct a binarized tile matrix and then performs iterative LSI on the matrix to extract meaningful low-dimensional representations. Similar to "Signac_by_cluster_peaks" approach, "ArchR_peaks" method first uses the latent representation obtained from "ArchR_tiles" for clustering, then performs peak calling per individual clusters and generates a consensus peak set by merging these peak tracks. Afterwards, iterative LSI is performed on the peak count matric.

During the iterative LSI process, at each iteration, the top accessible features (in 1st iteration) or top variable features (since 2nd iteration) are selected for LSI. The resulting cell clusters are then identified and utilized for feature selection in the subsequent iteration, enabling an iterative refinement of the LSI procedure.

We used the R package ArchR(v1.0.3) for implementation. When running the function `addIterativeLSI()`, Louvain algorithm was used for the clustering in intermediate steps with increasing resolutions, and no subsampling of cells was performed. Other parameters we used were set to be the default values.

SnapATAC

The SnapATAC method (version 1) takes the fragment files as input and first constructs a binary cell-by-bin matrix using 5000-bp non-overlapping genomic bins. Then, after filtering out unwanted bins, it computes a pairwise cell-to-cell similarity matrix using Jaccard coefficient. This kernel matrix is subject to normalization of the coverage bias and then eigenvalue decomposition (EVD) to get the cell embeddings.

For the implementation, we used the command line tool snaptools (v1.4.8) to create snap files from fragment files, and the R package SnapATAC (v1.0.0) for the rest of the processing pipeline. We followed the standard procedures in https://github.com/r3fang/SnapATAC/tree/master/examples/10X_PBMC_15K, except that we ran the function `runDiffusionMaps()` using all cells instead of using landmark cells. This approach was chosen in order to maintain a consistent basis for comparison with other methodologies. Furthermore, the datasets employed are of small to moderate size, and running all

methods using all cells does not pose significant efficiency issues, which is the primary concern that subsampling procedures are designed to address.

SnapATAC2

SnapATAC2 is the version 2 of SnapATAC method and it is released as a python package. By implementing AnnData object and optimizing the on-disk representation, it facilitates the processing of high-dimensional data. As demonstrated in the tutorial https://kzhang.org/SnapATAC2/tutorials/pbmc.html, SnapATAC2 first creates a cell-by-bin matrix containing insertion counts using 500-bp bins by default. Then, a pairwise cell-to-cell similarity matrix is generated, using either Jaccard coefficient (SnapATAC2_jaccard) or cosine similarity (SnapATAC2_cosine). With this kernel matrix, the symmetric normalized graph Laplacian is computed, and the bottom eigenvectors of the graph Laplacian is used as the lower dimensional representation. For implementation, we used SnapATAC2 (v2.2.0). To select features, we removed bins overlapping with the blacklist regions as always done in other methods and called function `snapatac2.pp.select_features()` with parameters `min_cells=10, most_variable=1000000`.

Clustering

In this study, we used a well-established graph-based clustering method for all clustering analyses. We first constructed a shared nearest neighbor graph and then applied the Leiden algorithm [20] using modularity as the optimization objective, as implemented in the Seurat package (v4.3.0) [36]. The Leiden algorithm incorporates a step where node partitions are refined by randomly reassigning nodes to communities that increase the objective function, enabling a wider exploration of the partition space [20]. To account for the inherent stochasticity in the Leiden algorithm, we ran it with 5 different random seeds: 0, 2, 5, 42, and 123. Since the optimal number of clusters is not known a priori, a range of resolutions was used to obtain diverse clustering solutions yielding varying numbers of clusters. The parameters we used to achieve the optimal solution for each method and dataset are presented in the Additional file 2.

Evaluation metrics

According to the data structure our evaluation applied to, we have classified our evaluation metrics into three categories: embedding-based, graph-based and partition-based (Fig. 1).

ASW, FNS

The silhouette width quantifies the average distance between an observation and the other observations within its cluster, relative to the average distance to the nearest neighboring cluster [42]. The Average Silhouette Width (ASW) is calculated as the mean silhouette width across all observations within a cluster, providing insights into the compactness of the cluster and its separation from other clusters. ASW values range from -1 to 1, with 1 indicating dense and well-separated clusters, 0 representing clusters that overlap, and -1 indicating significant misclassification, where within-cluster dispersion is greater than between-cluster dispersion.

A limitation of ASW is that it is not invariant to the scaling of the space. As a solution, we introduced the fraction of negative Silhouette score (FNS) to assess the cluster-level proportion of cells with a negative Silhouette width. FNS characterizes the fraction of cells with a smaller distance to cells within another cluster compared to their own cluster. It is robust to linear scaling and enables more meaningful comparisons across different dimensional reduction methods.

cLISI

The LISI has been proposed to evaluate either the mixing between batches or the separation between cell types [43]. To calculate it, a weighted k-nearst neighbor (kNN) graph is first generated based on Euclidean distance within an embedding space. Subsequently, for each node in the graph, it computes the expected number of cells needed to be sampled before two cells are drawn from the same batch/clusters within its neighborhood. We used the cluster-based variant of LISI, known as cluster LISI (cLISI), as a metric to assess the embedding representation. This is implemented by using the function `compute_lisi()` from the R package lisi v1.0 [43, 44]. cLISI ranges from 1 to K, where K is the total number of cell types in the dataset. One indicates a neighborhood consisting exclusively of cells from a single cell type, while K corresponds to complete mixing, with cells from all cell types found within the neighborhood.

PWC

Partition-based metrics are susceptible to the influence of clustering parameters, whereas embedding-based metrics rely on the proper definition of similarity within the embedding space, which may not necessarily align with the similarity employed in clustering. Therefore, we proposed a novel graph-based metric that directly operates on the graph where cells of the same (ground truth) type are identified as communities. Filippo et al. [45] discussed a definition of community in the network by splitting the total degree of a node i into two contributions: given a subgraph $V \subset G$, $k_i(V) = k_i^{in}(V) + k_i^{out}(V)$, where $k_i^{in}(V)$ is the number of edges connecting node i to other nodes in V, and $k_i^{out}(V)$ is the number of connections towards the rest of the network. The subgraph V is a community in a strong sense if $k_i^{in}(V) > k_i^{out}(V), \forall i \in V$. Inspired by this definition, we introduced the metric Proportion of Weakly Connected cells (PWC). PWC quantifies, for a subgraph V consisting of all the cells of the same true class, the proportion of cells that have fewer connections within V than with the rest of the graph.

AW, AV

Wallace [46] proposed two asymmetric indices to quantify the similarity between two partitions of a set. Let $U = \{U_1, U_2, ..., U_I\}$ be the partition of the dataset defined by cell types and $Z = \{Z_1, Z_2, ..., Z_J\}$ be the partition given by the clustering prediction. The first index W is the proportion of joint object pairs in partition U that are also joined in partition Z. The second index V is the proportion of joint object pairs in partition Z that are also joined in partition U. Both index W and V can be adjusted for chance using formula:

$$\mathrm{AS} = \frac{\mathrm{S} - E(\mathrm{S})}{1 - E(\mathrm{S})}, \tag{1}$$

where S is a similarity measure that does not have value 0 under statistical independence. A generalized hypergeometric model is assumed to calculate the expectation value of V and W [47]. AW can be interpreted as the completeness of cell types. It quantifies to what extend objects belonging to the same cell type in U are assigned to the same cluster in Z. Similarly, AV can be interpreted as the homogeneity of clusters, which measures to what extend clusters are not mixing objects of different cell types. AW and AV can be decomposed into indices for the individual cell types of partitions U and for the individual clusters of partitions Z, respectively [48], that is:

$$\mathrm{AW}_i = \frac{N\sum_{j=1}^{J}\binom{n_{ij}}{2} - \binom{n_{i+}}{2}Q}{\binom{n_{i+}}{2}(N-Q)}, \tag{2}$$

$$\mathrm{AV}_j = \frac{N\sum_{i=1}^{I}\binom{n_{ij}}{2} - \binom{n_{+j}}{2}P}{\binom{n_{+j}}{2}(N-P)}, \tag{3}$$

where n_{ij} is the number of objects placed in class U_i and in cluster Z_j , N is the total number of pairs of objects, $P = \sum_{i=1}^{I}\binom{n_{i+}}{2}$ is the number of object pairs that were placed in the same cluster in U, and $Q = \sum_{j=1}^{J}\binom{n_{+j}}{2}$ is the number of object pairs that were placed in the same cluster in Z. AW and AV range from −1 to 1, with 0 for random assignments and 1 for perfect agreement.

ARI, ARI2

The adjusted Rand index is the harmonic mean of AW and AV, and therefore a summary of both the homogeneity of predicted clusters and the completeness of true classes. ARI can be decomposed into a weighted average of the AW_i's and AV_j's as follows [48]:

$$\mathrm{ARI} = \frac{\sum_{i=1}^{I}\mathrm{AW}_i P_i + \sum_{j=1}^{J}\mathrm{AV}_j Q_j}{\sum_{i=1}^{I}P_i + \sum_{j=1}^{J}Q_j}, \tag{4}$$

where $P_i = \binom{n_{i+}}{2}$ is the number of object pairs in cluster U_i, and $Q_j = \binom{n_{+j}}{2}$ is the number of object pairs in cluster Z_j. Equation 4 shows that ARI is largely determined by the AW_i and AV_j values of large clusters. However, in many cases in single-cell analysis, the rare cell types are of more concern. Therefore, we included a variant of ARI (ARI2) proposed by Matthijs et al. [48] to alleviate the class size bias of ARI.

$$\mathrm{ARI2} = \frac{2\mathrm{AW}' \times 2\mathrm{AV}'}{\mathrm{AW}' + \mathrm{AV}'}, \tag{5}$$

where

$$\mathrm{AW}' = \frac{1}{I}\sum_{i=1}^{I}\mathrm{AW}_i, \tag{6}$$

and

$$\mathrm{AV}' = \frac{1}{J}\sum_{j=1}^{J}\mathrm{AV}_j. \tag{7}$$

MI, VI

While ARI, AW, and AV are external evaluation metrics that count pairs of objects, Mutual Information (MI) and Variation of Information (VI) are based on information theory. These two groups of metrics do not always show consistent results, due to different underlying assumption. The MI between two partitions U and Z is as follows:

$$\mathrm{MI} = \sum_{i=1}^{I}\sum_{j=1}^{J} p_{ij} log \frac{p_{ij}}{p_i p_j}, \tag{8}$$

where $p_{ij} = \frac{n_{ij}}{n}$, $p_i = \frac{n_{i+}}{n}$, and $p_j = \frac{n_{+j}}{n}$. It has been shown [49] that

$$\mathrm{MI} = H(Z) - H(Z|U), \tag{9}$$

where $H(\cdot)$ is the Shannon entropy. Since Z stays the same for a given dataset, Eq. 9 indicates that comparing MI between methods on the same dataset is equivalent to comparing the conditional Shannon entropy of Z on U. In other words, MI can be interpreted as the measure of homogeneity of clusters, similar to AV. Note that MI is not normalized and the upper bound varies across datasets. It is therefore only meaningful to compare MI within the same dataset.

VI measures the amount of information that is lost or gained in changing from partition Z to U:

$$\mathrm{VI} = -\sum_{i=1}^{I} p_i log p_i - \sum_{j=1}^{J} p_j log p_j - 2\sum_{i=1}^{I}\sum_{j=1}^{J} p_{ij} log \frac{p_{ij}}{p_i p_j}, \tag{10}$$

Similarly, VI is also highly related to entropy:

$$\mathrm{VI} = H(U|Z) + H(Z|U). \tag{11}$$

VI is not normalized, and a higher VI value indicates a worse clustering solution.

Calculating and comparing the area under the curve

Partition-based metrics change as the clustering resolution changes. The true cluster numbers is not predetermined, and the optimal performance is not always achieved at the true number of clusters. Therefore, comparing clusterings at a fixed resolution or number of clusters becomes challenging. To address this challenge, we compared clusterings across a range of resolution parameters that result in varying number of clusters

(Additional file 1: Fig. S20a). We examined the performance as the number of clusters changes and summarized the results using the area under the curve (AUC).

To calculate the AUC and compare between results of different ranges of cluster numbers, the upper bound of each metric is used for the normalization of the absolute AUC. Specifically, metrics such as ARI, ARI2, AW, and AV have an upper bound of 1. MI and VI were normalized using the empirical maximum value per method per dataset. Notably, in the case of VI, instead of using the normalized AUC directly for comparison, we used 1− normalized AUC.

When plotting the AUC heatmap, we colored the heatmap using the deviations from the column-wise median scaled by matrix-wise median absolute deviation. Let $\boldsymbol{A}$ be the original matrix storing the metric values, $\boldsymbol{B}$ be the transformed matrix, and $A_{i,j}$, $B_{i,j}$ is the element of matrix $\boldsymbol{A}$, and $\boldsymbol{B}$, respectively. The calculation of $\boldsymbol{B}$ is as in Eqs. 12, 13, and 14. By applying this transformation, the color scale is unified across datasets.

$$B_{i,j} = \frac{A_{i,j} - \text{Median}(\boldsymbol{A}_{+j})}{\text{Median}(\boldsymbol{M'})}, \tag{12}$$

where

$$M'_{i,j} = |M_{i,j}|, \tag{13}$$

and

$$\boldsymbol{M}_{+j} = \boldsymbol{A}_{+j} - \text{Median}(\boldsymbol{A}_{+j}). \tag{14}$$

Choosing the number of dimensions

When applying dimensional reduction methods, a parameter that one needs to choose is the number of dimensions n of the embedding space to use. Since all the methods use either principle component analysis (PCA) or singular value decomposition (SVD), we applied the elbow approach and examined the scree plot of each method by plotting the proportion of variance explained by each component against the component indices. We observed that for nearly all methods and datasets, the elbow point is before 15 dimensions (Additional file 1: Fig. S20b). We therefore used $n = 15$ for all methods. More details on how n affects the performance is discussed in the "Results" section.

Analysis of feature numbers

To understand how the feature numbers impact the performance, we run each method across a range of feature numbers: 25k, 100k, 200k, and 500k. For ArchR and SnapATAC2, adjusting these settings was straightforward, achieved by changing the `varFeatures`argument in the `addIterativeLSI()` function, or the `most_variable` argument in the `snapatac2.pp.select_features()` function, respectively. In the case of Signac and SnapATAC, retaining the desired number of features involves computing the quantiles. The calculated quantile was specified through the argument `min.cutoff` in the function `FindTopFeatures()` in Signac. For SnapATAC, after initially removing the top 5% most accessible (and thus least variable) features, a second

filtering step was done by selecting the top n% most accessible features. Here, n% was determined to ensure the retention of the specified feature counts. Then, the clustering was performed using the same optimal hyper-parameters as in the previous analysis which used the default feature numbers.

Analysis of clustering robustness

As described in the Clustering section, we performed Leiden clustering using a range of resolution values in combination with 5 random seeds. For each resolution, we compared the clustering outcomes from different seed pairs by computing ARI. An ARI of 1 indivates identical clustering predictions from the two seeds, whereas an ARI of 0 suggests that the clusterings are independent. We then calculated the deviation of the pairwise ARI from 1, interpreting this as a measure of the clustering's robustness when fixing resolutions. To get an overall estimation of robustness as in Additional file 1: Fig. S12a, this deviation was averaged across seed pairs and resolutions.

Analysis of library size biases

The library size of each cell is the total number of unique fragments of that cell in the fragment file. To quantify to what extent the learned latent space is driven by the library size, we calculated an absolute Pearson's correlation coefficient value $r_{l,d}$ for each latent dimension l and dataset d as follows:

$$r_{l,d} = \frac{\sum_k |\text{cor}(\text{sqrt}(\boldsymbol{c}_{k,d}), \boldsymbol{x}_{k,l,d})|}{K}, \tag{15}$$

where cor() is the function to calculate Pearson's correlation coefficient between two vectors, and sqrt() is the function to calculate element-wise square root of a vector. $\boldsymbol{c}_{k,d}$ represents the library size across all cells of cell type k in dataset d, and $\boldsymbol{x}_{k,l,d}$ represents the value of the latent component l across all cells of cell type k in dataset d. K is the total number of cell types. $r_{l,d}$ for dataset 10XPBMC and $l = 1, 2, 3, 4, 5$ are shown in Additional file 1: Fig. S14.

To summarize across methods, we averaged this value across $l = 1, 2, 3, 4, 5$ and the six datasets (see Fig. 5).

Benchmarking methods for predicting gene activity scores

From the five methods discussed, four offer the capability to infer gene activities. Signac employs a basic technique, counting fragments in the gene body and promoter regions for each gene, followed by log-normalization of these counts to derive a gene activity score. SnapATAC calculates gene body fragment counts, normalizes them using log-transformed count-per-million reads, and then employs a Markov affinity-graph-based method for imputation and smoothing. SnapATAC2 has a similar procedure to SnapATAC, with the distinction of counting TN5 insertions instead of fragments. ArchR adopts a more complex model that weights fragment counts based on their distance to the Transcription Start Site (TSS) and includes distal regulatory elements, while also considering neighboring genes.

To benchmark their performance on predicting gene activities, we utilized the two multi-omics datasets, namely 10XPBMC and Chen2019. Gene expression data from the

RNA components were used as the ground truth. Additionally, we identified 500 metacells per dataset using k-means clustering, applied through the function fastcluster(return Type="metacells") in scDblFinder (v 1.13.9). We then selected the top 1000 highly variable genes and computed Pearson's and Kendall's correlation coefficients to compare the predicted gene activity scores with the actual gene expression data. Specifically, we computed both per-cell or metacell correlation across genes as well as per-gene correlation across cell or metacell.

Benchmarking methods for integration with scRNA-seq data

The cross-modality integration was performed in an unpaired fassion, meaning we treated the cells from two modalities as if they were unmatched. Specifically, we used GLUE [50] for such integration, where the processing of RNA are fixed, and the features and cell embeddings generated using different ATAC-seq processing pipelines are used as GLUE's input. For the construction of the guidence graph in GLUE, we used the top 2000 high variable genes (HVGs) from the RNA data and the genomic features from ATAC data that were (i) selected by the processing method for dimensional reduction and (ii) are connected to any HVGs (i.e., overlapping in either the gene body or promoter region).

FOSCTTM are calculated as follows: let X_1 and X_2 be the cell embeddings of RNA and ATAC data within an integrated space, respectively. Each matrix is of size $n \times m$, where n represents the number of cells and m is the number of latent dimensions. The rows within X_1 and X_2 are aligned such that each corresponds to the same cell. Define $d_{i,j}$ as the cosine distance between $X_1[i,:]$ and $X_2[j,:]$. For a given cell i, the fraction of samples closer than its true match is computed as:

$$\text{frac}_i = \frac{R_i - 1}{n - 1} \tag{16}$$

, where R_i is the rank of $d_{i,i}$ among the distances $d_{i,1}, d_{i,2}, ..., d_{i,n}$ that are less than or equal to $d_{i,i}$. Then, the frac_i were also calculated after switching X_1 and X_2, and then averaged for the cell pair i.

Other indices

Evenness

Evenness (E) quantifies the homogeneity of abundances of different types in a sample [51]. Here, we use Eq. 17 to calculate E:

$$\text{E} = \frac{\exp(H(U))}{K}, \tag{17}$$

where $H(\cdot)$ is Shannon entropy, and K is the total number of cell types. E ranges from $\frac{1}{K}$ to 1, and a higher E indicates that the dataset is more balanced.

Geary's C

We calculated Geary's C index [52] of log-transformed fragment counts using spatial distance defined by k-nearest neighbor (KNN) graph ($k = 20$) [14]. Geary's C is calculated as:

$$C = \frac{(N-1)\sum_i \sum_j w_{ij}(x_i - x_j)^2}{2S_0 \sum i(x_i - \bar{x}^2)}, \tag{18}$$

where N is the total number of cells; x_i is the log-transformed fragment counts of cell i, w_ij is the weight of edge between cell i and j on the KNN graph, and S_0 is the sum of all weights in $\boldsymbol{W}$.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13059-024-03356-x.

Additional file 1. Supplementary figures S1-21

Additional file 2. Supplementary tables S1-3

Additional file 3. Review history

Acknowledgements
We thank all members of the von Meyenn group and Robinson group for helpful discussions and support. We specifically thank Adhideb Ghosh for advice on the study design and scRNA-seq data analysis, João Pedro Agostinho de Sousa for help with setting up the computational environments, and Emanuel Sonder for his feedbacks on the manuscript draft.

Peer review information
Andrew Cosgrove was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Review history
The review history is available as Additional file 3.

Authors' contributions
SL and FvM conceptualized the study with the help from MDR and PLG. SL designed the benchmark, prepared the data, wrote the code for the pipeline, and performed the analysis. SL, PLG, MDR, and FvM interpreted the results. SL wrote the manuscript draft. All authors reviewed and approved the final version of this manuscript.

Funding
Open access funding provided by Swiss Federal Institute of Technology Zurich This work was supported by ETH Zurich core funding (FvM) and UZH core funding (MDR).

Availability of data and materials
Our benchmarking workflow is provided as a reproducible Snakemake pipeline on GitHub: https://github.com/RoseYuan/sc_chromatin_benchmark [53]. Notebooks, R scripts, and supporting data used for preprocessing datasets and generating all the visualizations in this manuscript is available at https://github.com/RoseYuan/benchmark_paper [38] ((snapshot on Zenodo [54]). For the analyzed datasets, the preprocessed data that can be directly input into the Snakemake pipeline is available on Zenodo [55]. For the unprocessed data, fragment files of the cell line dataset were downloaded from GEO accession GSE162690 [56], fragment file and the gene expression matrix file of the 10X PBMC multiomics dataset were downloaded from https://www.10xgenomics.com/resources/datasets/pbmc-from-a-healthy-donor-granulocytes-removed-through-cell-sorting-10-k-1-standard-1-0-0, and fragment files of the human adult atlas datasets were downloaded from GEO accession GSE184462 [57]. Bam files of the dataset Buenrostro2018 was downloaded from GEO accession GSE96772 [58]. Bam file of the ATAC part of Chen2019 dataset was processed by Stuart's lab, and we downloaded the fragment files they provided at https://stuartlab.org/signac/articles/snareseq.html, and the RNA part was downloaded from GEO accession GSE126074 [59]. The Seurat objects of the Allen mouse brain reference dataset used for annotating the scRNA-seq data were downloaded from Signac's website: https://signac-objects.s3.amazonaws.com/allen_brain.rds. The PBMC reference dataset was downloaded here: https://atlas.fredhutch.org/data/nygc/multimodal/pbmc_multimodal.h5seurat.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 4 August 2023 Accepted: 29 July 2024
Published online: 16 August 2024

References

1. Buenrostro JD, Wu B, Litzenburger UM, Ruff D, Gonzales ML, Snyder MP, et al. Single-cell chromatin accessibility reveals principles of regulatory variation. Nature. 2015;523(7561):486–90.
2. Cusanovich DA, Daza R, Adey A, Pliner HA, Christiansen L, Gunderson KL, et al. Multiplex single-cell profiling of chromatin accessibility by combinatorial cellular indexing. Science. 2015;348(6237):910–4.
3. Kaya-Okur HS, Wu SJ, Codomo CA, Pledger ES, Bryson TD, Henikoff JG, et al. CUT &Tag for efficient epigenomic profiling of small samples and single cells. Nat Commun. 2019;10(1):1930.
4. Chen H, Lareau C, Andreani T, Vinyard ME, Garcia SP, Clement K, et al. Assessment of computational methods for the analysis of single-cell ATAC-seq data. Genome Biol. 2019;20(1):1–25.
5. Moon KR, Stanley JS III, Burkhardt D, van Dijk D, Wolf G, Krishnaswamy S. Manifold learning-based methods for analyzing single-cell RNA-sequencing data. Curr Opin Syst Biol. 2018;7:36–46.
6. Wagner DE, Klein AM. Lineage tracing meets single-cell omics: opportunities and challenges. Nat Rev Genet. 2020;21(7):410–27.
7. Rautenstrauch P, Vlot AHC, Saran S, Ohler U. Intricacies of single-cell multi-omics data integration. Trends Genet. 2022;38(2):128–39.
8. Stuart T, Srivastava A, Madad S, Lareau CA, Satija R. Single-cell chromatin state analysis with Signac. Nat Methods. 2021;18(11):1333–41.
9. Granja JM, Corces MR, Pierce SE, Bagdatli ST, Choudhry H, Chang HY, et al. ArchR is a scalable software package for integrative single-cell chromatin accessibility analysis. Nat Genet. 2021;53(3):403–11.
10. Fang R, Preissl S, Li Y, Hou X, Lucero J, Wang X, et al. Comprehensive analysis of single cell ATAC-seq data with SnapATAC. Nat Commun. 2021;12(1):1337.
11. de Boer CG, Regev A. BROCKMAN: deciphering variance in epigenomic regulators by k-mer factorization. BMC Bioinformatics. 2018;19(1):1–13.
12. Ji Z, Zhou W, Ji H. Single-cell regulome data analysis by SCRAT. Bioinformatics. 2017;33(18):2930–2.
13. Pliner HA, Packer JS, McFaline-Figueroa JL, Cusanovich DA, Daza RM, Aghamirzaie D, et al. Cicero predicts cis-regulatory DNA interactions from single-cell chromatin accessibility data. Mol Cell. 2018;71(5):858–71.
14. Ashuach T, Reidenbach DA, Gayoso A, Yosef N. PeakVI: a deep generative model for single-cell chromatin accessibility analysis. Cell Rep Methods. 2022;2(3):100182.
15. Yuan H, Kelley DR. scBasset: sequence-based modeling of single-cell ATAC-seq using convolutional neural networks. Nat Methods. 2022;19(9):1088–96.
16. Tayyebi Z, Pine AR, Leslie CS. Scalable and unbiased sequence-informed embedding of single-cell ATAC-seq data with CellSpace. Nat Methods. 2024;21:1014–22. https://doi.org/10.1038/s41592-024-02274-x.
17. Cusanovich DA, Hill AJ, Aghamirzaie D, Daza RM, Pliner HA, Berletch JB, et al. A single-cell atlas of in vivo mammalian chromatin accessibility. Cell. 2018;174(5):1309–24.
18. Weber LM, Saelens W, Cannoodt R, Soneson C, Hapfelmeier A, Gardner PP, et al. Essential guidelines for computational method benchmarking. Genome Biol. 2019;20:1–12.
19. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019;15(6):e8746.
20. Traag VA, Waltman L, Van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep. 2019;9(1):5233.
21. Klamann C, Lau C, Schwartz GW. TooManyCellsInteractive: a visualization tool for dynamic exploration of single-cell data. bioRxiv. 2023:2023–06.
22. Mishra S, Monath N, Boratko M, Kobren A, McCallum A. An evaluative measure of clustering methods incorporating hyperparameter sensitivity. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36. 2022. pp. 7788–7796. https://doi.org/10.1609/aaai.v36i7.20747.
23. Germain PL, Sonrel A, Robinson MD. pipeComp, a general framework for the evaluation of computational pipelines, reveals performant single cell RNA-seq preprocessing tools. Genome Biol. 2020;21(1):1–28.
24. Germain PL, Lun A, Garcia Meixide C, et al. Doublet identification in single-cell sequencing data using scDblFinder [version 2; peer review: 2 approved]. F1000Research. 2022;10:979. https://doi.org/10.12688/f1000research.73600.2.
25. Zhang K, Hocker JD, Miller M, Hou X, Chiou J, Poirion OB, et al. A single-cell atlas of chromatin accessibility in the human genome. Cell. 2021;184(24):5985–6001.
26. Buenrostro JD, Corces MR, Lareau CA, Wu B, Schep AN, Aryee MJ, et al. Integrated single-cell analysis maps the continuous regulatory landscape of human hematopoietic differentiation. Cell. 2018;173(6):1535–48.
27. Chen S, Lake BB, Zhang K. High-throughput sequencing of the transcriptome and chromatin accessibility in the same cell. Nat Biotechnol. 2019;37(12):1452–7.
28. Miao Z, Kim J. Is single nucleus ATAC-seq accessibility a qualitative or quantitative measurement? bioRxiv. 2022:2022–04.
29. Hill A. Dimensionality reduction for scATAC data. 2019. http://andrewjohnhill.com/blog/2019/05/06/dimensionality-reduction-for-scatac-data/. Accessed 12 July 2023.

30. Ortega A, Frossard P, Kovačević J, Moura JM, Vandergheynst P. Graph signal processing: overview, challenges, and applications. Proc IEEE. 2018;106(5):808–28.
31. Liu J, Huang Y, Singh R, Vert JP, Noble WS. Jointly embedding multiple single-cell omics measurements. In: Algorithms in bioinformatics:... International Workshop, WABI..., proceedings. WABI (Workshop), vol. 143. NIH Public Access; 2019.
32. Ahlmann-Eltze C, Huber W. Comparison of transformations for single-cell RNA-seq data. Nat Methods. 2023;20:1–8.
33. Heumos L, Schaar AC, Lance C, Litinetskaya A, Drost F, Zappia L, et al. Best practices for single-cell analysis across modalities. Nat Rev Genet. 2023;24:1–23.
34. Martens LD, Fischer DS, Theis FJ, Gagneur J. Modeling fragment counts improves single-cell ATAC-seq analysis. bioRxiv. 2022;21:2022–05.
35. Tim Stuart WWK. Sinto: single-cell analysis tools. GitHub; 2019. https://github.com/timoast/sinto.
36. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, et al. Comprehensive integration of single-cell data. Cell. 2019;177(7):1888–902.
37. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech Theory Exp. 2008;2008(10):P10008.
38. Luo S, Germain PL, Robinson MD, von Meyenn F. Code and data for the manuscript "Benchmarking computational methods for single-cell chromatin data analysis". GitHub; 2023. https://github.com/RoseYuan/benchmark_paper.
39. Zappia L, Oshlack A. Clustering trees: a visualization for evaluating clusterings at multiple resolutions. Gigascience. 2018;7(7):giy083.
40. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;184(13):3573–87.
41. Lein ES, Hawrylycz MJ, Ao N, Ayres M, Bensinger A, Bernard A, et al. Genome-wide atlas of gene expression in the adult mouse brain. Nature. 2007;445(7124):168–76.
42. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987;20:53–65.
43. Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, et al. Fast, sensitive and accurate integration of single-cell data with Harmony. Nat Methods. 2019;16(12):1289–96.
44. Ilya Korsunsky KS. Methods to compute Local Inverse Simpson's Index (LISI). GitHub; 2019. https://github.com/immunogenomics/LISI.
45. Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D. Defining and identifying communities in networks. Proc Natl Acad Sci. 2004;101(9):2658–63.
46. Wallace DL. A method for comparing two hierarchical clusterings: comment. J Am Stat Assoc. 1983;78(383):569–76.
47. Severiano A, Pinto FR, Ramirez M, Carriço JA. Adjusted Wallace coefficient as a measure of congruence between typing methods. J Clin Microbiol. 2011;49(11):3997–4000.
48. Warrens MJ, van der Hoef H. Understanding the adjusted Rand index and other partition comparison indices based on counting object pairs. J Classif. 2022;39(3):487–509.
49. Wu J, Chen J, Xiong H, Xie M. External validation measures for K-means clustering: a data distribution perspective. Expert Syst Appl. 2009;36(3):6050–61.
50. Cao ZJ, Gao G. Multi-omics single-cell data integration and regulatory inference with graph-linked embedding. Nat Biotechnol. 2022;40(10):1458–66.
51. Hill MO. Diversity and evenness: a unifying notation and its consequences. Ecology. 1973;54(2):427–32.
52. Geary RC. The contiguity ratio and statistical mapping. Inc Stat. 1954;5(3):115–46.
53. Luo S, Germain PL, Robinson MD, von Meyenn F. Snakemake workflow to benchmark computational methods for single-cell chromatin data analysis. GitHub; 2023. https://github.com/RoseYuan/sc_chromatin_benchmark.
54. Luo S, Germain PL, Robinson MD, von Meyenn F. Code and data for the manuscript "Benchmarking computational methods for single-cell chromatin data analysis". Zenodo; 2024. https://doi.org/10.5281/zenodo.12607316.
55. Luo S, Germain PL, Robinson MD, von Meyenn F. Data for the manuscript "Benchmarking computational methods for single-cell chromatin data analysis". Zenodo; 2023. https://doi.org/10.5281/zenodo.8212920.
56. Granja JM, Corces MR. ArchR: An integrative and scalable software package for single-cell chromatin accessibility analysis. Gene Expression Omnibus; 2020. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE162690.
57. Zhang K, Hocker JD, Miller M, Hou X, Poirion OB, Wang A, et al.. A single-cell atlas of chromatin accessibility in the human genome. Gene Expression Omnibus; 2021. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE184462.
58. Buenrostro JD, Corces MR, Lareau CA, Wu B, Schep AN, Aryee MJ, et al.. Single-cell epigenomics maps the continuous regulatory landscape of human hematopoietic differentiation. Gene Expression Omnibus; 2018. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE96772.
59. Chen S, Zhang K. Simultaneous profiling of transcriptome and chromatin accessibility in single nucleus. Gene Expression Omnibus; 2019. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE126074.

Publisher's Note