PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Scedar: A scalable Python package for single-cell RNA-seq exploratory data analysis

Yuanchao Zhang[1,2], **Man S. Kim**[1], **Erin R. Reichenberger**[1], **Ben Stear**[1], **Deanne M. Taylor**[1,3]*

1 Department of Biomedical and Health Informatics, The Children's Hospital of Philadelphia, Philadelphia, Pennsylvania, United States of America, **2** Department of Genetics, Rutgers University, Piscataway, New Jersey, United States of America, **3** Department of Pediatrics, Perelman School of Medicine, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America

* taylordm@email.chop.edu

Citation: Zhang Y, Kim MS, Reichenberger ER, Stear B, Taylor DM (2020) Scedar: A scalable Python package for single-cell RNA-seq exploratory data analysis. PLoS Comput Biol 16(4): e1007794. https://doi.org/10.1371/journal.pcbi.1007794

Editor: Mihaela Pertea, Johns Hopkins University, UNITED STATES

Received: April 8, 2019

Accepted: March 17, 2020

Published: April 27, 2020

Data Availability Statement: Used for scedar benchmarking, the real scRNA-seq datasets can be obtained from the following websites: https://hemberg-lab.github.io/scRNA.seq.datasets/, https://support.10xgenomics.com/single-cell-gene-expression/datasets, https://figshare.com/articles/MCA_DGE_Data/5435866, and https://oncoscape.v3.sttrcancer.org/atlas.gs.washington.edu.mouse.rna/downloads. The simulated datasets can be generated using scripts at https://github.com/TaylorResearchLab/scedar/tree/master/docs/r_scripts.

Abstract

In single-cell RNA-seq (scRNA-seq) experiments, the number of individual cells has increased exponentially, and the sequencing depth of each cell has decreased significantly. As a result, analyzing scRNA-seq data requires extensive considerations of program efficiency and method selection. In order to reduce the complexity of scRNA-seq data analysis, we present scedar, a scalable Python package for scRNA-seq exploratory data analysis. The package provides a convenient and reliable interface for performing visualization, imputation of gene dropouts, detection of rare transcriptomic profiles, and clustering on large-scale scRNA-seq datasets. The analytical methods are efficient, and they also do not assume that the data follow certain statistical distributions. The package is extensible and modular, which would facilitate the further development of functionalities for future requirements with the open-source development community. The scedar package is distributed under the terms of the MIT license at https://pypi.org/project/scedar.

This is a *PLOS Computational Biology* Software paper.

Introduction

Cost-effective large-scale transcriptomic profiling of individual cells is enabled by the development of microfluidic, nanodroplet, and massively parallel sequencing technologies. Using these technologies, single-cell RNA-seq (scRNA-seq) experiments usually generate transcriptomic profiles of thousands to millions of individual cells [1]. Therefore, scRNA-seq has become more commonly used to either study specific biological questions or comprehensively profile certain tissues or organisms [2–4].

Analyses of scRNA-seq datasets therefore require efficient computational programs and sophisticated statistical methods. The programs should be able to manage memory efficiently, exploit multiple cores of the processing units, and handle errors and exceptions gracefully. The statistical methods must be able to function against high dimensionality, low signal-to-noise ratio, and different characteristics of data generated from different technologies and protocols

Funding: All authors received support for this work from the Department of Biomedical and Health Informatics and the CHOP Research Institute at The Children's Hospital of Philadelphia. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

[5–7]. Such requirements can become a barrier between experimental design and biological interpretations of the results.

In order to be scalable, methods have been designed to minimize the usage of hardware resources, so that a large-scale scRNA-seq dataset can be analyzed using a desktop computer, such as Seurat v3.0 [8] and Scanpy [9]. Seurat is an R package providing visualization and robust statistical methods to explore and interpret the heterogeneity of the dataset. Scanpy is a Python package providing efficient reimplementations of pre-existing statistical methods and analytical workflows, which can be used to perform general exploratory data analysis and special inferences.

Here, we attempt to achieve the scalability of scRNA-seq data analysis through an alternative approach, which is to enable the user to exploit powerful analytical methods using modern high-performance computing architectures [10–12], such as servers and clusters with large amount of memory, multiple central and graphical processing unit cores, and solid-state drives (SSD) with higher read and write speed than traditional hard disk drive (HDD). Such computational resources have been made easily accessible by cloud computing services like Amazon Web Services, Google Cloud, and Microsoft Azure. Applying this approach, we designed the analytical methods to be able to run in parallel processes with minimized memory overhead.

Additionally, we also aim to develop robust exploratory data analysis (EDA) methods, so that they could be applied to various datasets generated by different experimental designs, technologies, and protocols. Single-cell RNA-seq datasets have distinct statistical characteristics if generated from different scRNA-seq technologies and platforms [6,7,13], such as SMARTer [14], Drop-seq [15] and 10x Genomics [16]. In order to avoid assumptions on the statistical distributions of the datasets, we incorporated efficient implementations of machine learning methods into the data exploration process. Through extensive exploration, the statistical properties of the data could be observed and used to guide the selection of appropriate methods for biological interpretation [17].

Therefore, we developed a scalable and reliable Python package, **s**ingle-**c**ell **e**xploratory **d**ata analysis for **R**NA-seq (scedar), to facilitate the exploration of large-scale scRNA-seq datasets. Scedar provides analytical routines for visualization, gene dropout imputation, rare transcriptomic profile detection, clustering, and identification of cluster separating genes. The visualization methods are integrated with the efficient scRNA-seq data structures to provide intuitive, convenient, and flexible plotting interfaces. We implemented methods to impute gene dropouts (Algorithm A in S1 Text) and detect rare transcriptomic profiles (Algorithm B in S1 Text) based on the k-nearest neighbor (KNN) algorithm. The detected rare transcriptomic profiles could be compared with their nearest neighbors in detail to identify rare cell states or types. For clustering analysis, we provide a novel cell clustering algorithm: Minimum Description Length (MDL) Iteratively Regularized Agglomerative Clustering (MIRAC), shown in Algorithm 1. To identify genes that distinguish cell clusters we provide a method utilizing XGBoost, a sparsity-aware gradient boosted tree system [18].

Methods

Scedar package design

We designed scedar in an object-oriented manner for quickly exploring large-scale scRNA-seq transcription level matrices on a remote server utilizing parallel computing techniques, in order to provide a robust and extensible platform for EDA, rather than surpassing the analytical performance of any of $\geq$ 275 existing scRNA-seq data analysis methods [17]. The application programming interface (API) is designed to be intuitive to users familiar with the R programming language. The standard analysis workflow is to explore the dataset, cluster cells,

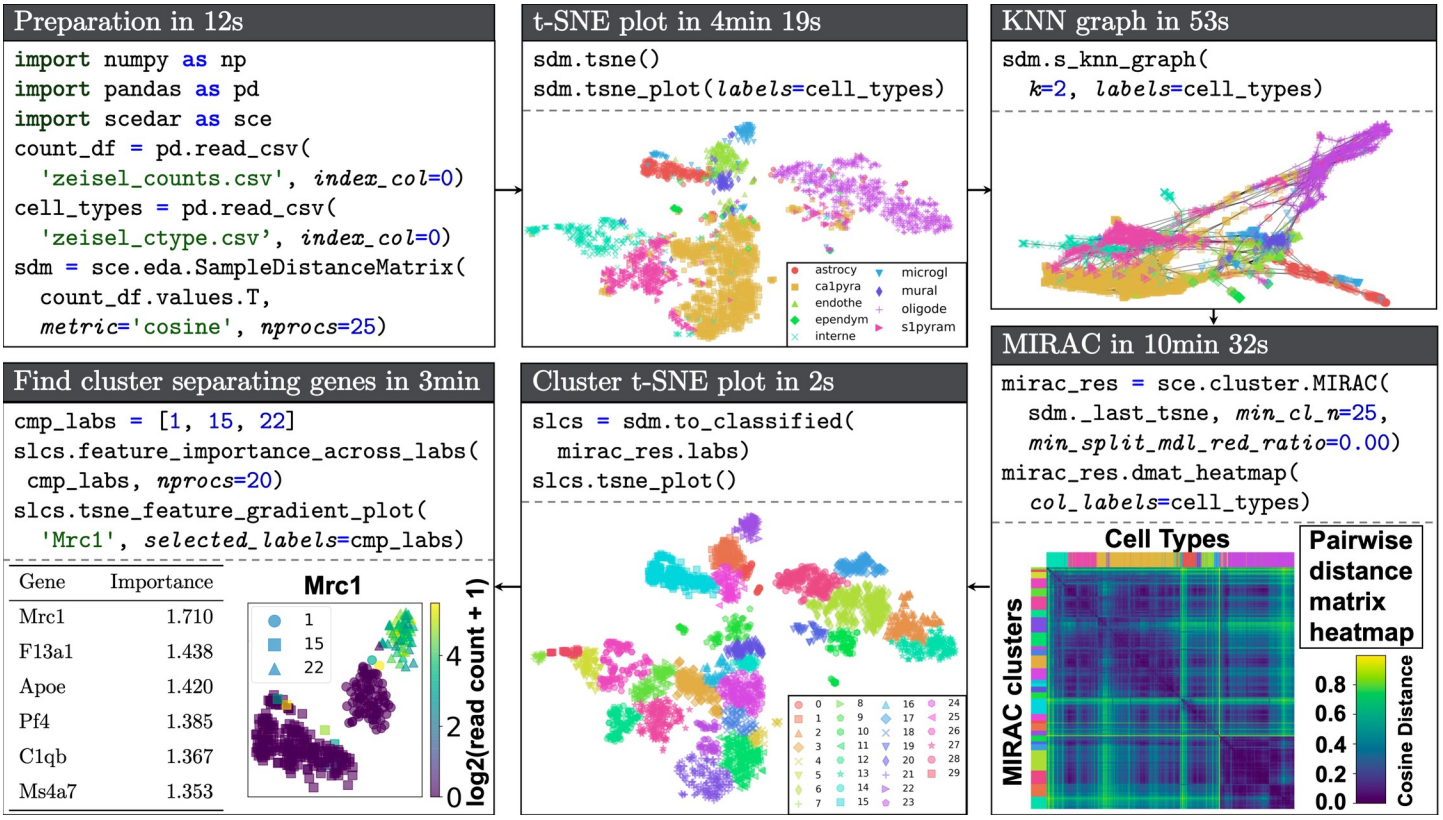

Fig 1. Demonstration workflow using scedar to analyze an scRNA-seq dataset with 3005 mouse brain cells and 19,972 genes generated using the STRT-Seq UMI protocol by Zeisel *et al.* [53]. Procedures and parameters that are not directly related to data analysis are omitted. The full version of the demo is available at https://github.com/logstar/scedar/tree/master/docs/notebooks.

https://doi.org/10.1371/journal.pcbi.1007794.g001

and identify cluster separating genes (Fig 1), which is implemented as four main modules: data structure, KNN, clustering and visualization.

The core data structure stores each transcription level matrix as a standalone sparse matrix or full array instance, and it can easily be extended to support customized analytical procedures. The built-in extensions include common procedures like pairwise distance computation, Principal Component Analysis (PCA), t-SNE [19], UMAP [20], and k-nearest neighbor graph [21]. We optimized time and memory efficiency with the following design patterns: parallel computing, lazy loading, caching and copy-on-write.

The KNN and clustering modules utilize the data structure and parallel computing to efficiently perform analytical procedures, and the results are stored in the data structure for further reference.

The visualization module contains plotting methods optimized for large datasets, especially for plotting heatmaps. For example, it takes less than two minutes to generate a heatmap image from a 50,000 x 20,000 matrix containing random standard normal entries.

Preprocessing is implemented as selection and transformation routines of the core data structure, which is not a focus of scedar. The package is designed to identify the necessity of certain preprocessing procedures by extensively exploring the original state of the data. Although preprocessing, such as batch effect correction and normalization, could facilitate the detection of biological variances between cells, applying preprocessing methods correctly requires careful validation of their assumptions, otherwise they may introduce unwanted bias or variability [22].

Minimum description length iteratively regulated agglomerative clustering

Minimum description length (MDL) iteratively regulated agglomerative clustering (MIRAC) extends hierarchical agglomerative clustering (HAC) [23] in a divide and conquer manner for scRNA-seq data. Input with raw or dimensionality reduced scRNA-seq data, MIRAC starts with one round of bottom-up HAC to build a tree with optimal linear leaf ordering [24], and the tree is then divided into small sub-clusters, which are further merged iteratively into clusters. Because each individual cluster becomes more homogenous with higher number of clusters, the iterative merging process is regularized with the MDL principle [25]. The asymptotic time complexity of the MIRAC algorithm is $O(n^4+mn^2)$ where n is the number of samples, and m is the number of features. The space complexity is $O(n^2+mn)$. Relevant mathematical theories and notations of MDL are briefly described in the following section. The pseudo-code of MIRAC is shown in Fig 2.

Comparing to HAC, MIRAC is not designed to be faster but rather to improve the cluster robustness. The asymptotic time complexity of MIRAC is the same as HAC with optimal leaf ordering. However, the use of MDL rather than deterministic similarity metrics, improves the noise tolerance by estimating similarity with probabilistic models to give more weight on signal and less weight on noise, assuming that the signal is stronger than noise.

Rationale. The rationale behind MIRAC is to reduce the number of cell partitions that need to be evaluated and find a good one among them.

The number of all possible partitions of a set with n elements is the Bell number, which could be computed with the following recurrence

$$B_n = \sum_{k=0}^{n-1} B_k \binom{n-1}{k}$$

where $n \geq 1$ and $B_0 = 1$ [26]. It is computationally intractable to compute the code lengths of B_n partitions, so we reduced the number of cell partitions to evaluate with the following steps:

1. We only evaluate the partitions of n cells that consecutively divide the optimal HAC tree T leaf ordering. The HAC tree with optimal leaf ordering maximizes the sum of similarity measurements between adjacent items in the leaf ordering [24], which could be computed with an algorithm developed by Bar-Joseph *et al.* that runs in time $O(n^4)$. The number of partitions that consecutively divide an ordering is the same as the number of compositions of n, which is $C_n = 2^{n-1}$. Because this number still grows exponentially with regard to n, it is necessary to further reduce the set of partitions for evaluation.

2. Within step 1 partitions, we only evaluate those with all clusters having $\geq t$ cells. The number of such partitions is the same as the compositions of n with all summands $\geq t$, which is

$$C_n^{\{t,t+1,\ldots,n\}} = \sum_{k=1}^{\lfloor n/t \rfloor} \binom{n-k(t-1)-1}{k-1}$$

given by [27]. The growth rate of $C_n^{\{t,t+1,\ldots,n\}}$ with regard to n is smaller, but we still need to reduce it further. For example, $C_n^{\{20,21,\ldots,n\}}$ for n in an $\langle 20, 40, 60, 80, 100, 120, 140, 160, 180, 200\rangle$ are $\langle 1, 2, 23, 274, 2695, 24941, 232016, 2184520, 20628613, 194570810\rangle$, and the values of 2^n are $\langle 1048576, 1099511627776, \ldots, 1.607 \text{ x } 10^{60}\rangle$.

3. Within step 2 partitions, we only evaluate the ones that could be generated by merging adjacent clusters of a specific partition $P^s = \{P_1^s, P_2^s, \ldots, P_k^s\}$, where P^s is generated by recursively dividing the root HAC tree T until the partitioned subtree has $\leq t-1$ leaves. Thus, $\lceil n/(t-1)\rceil \leq k \leq n$. Let $M(P^s)$ denote the number of step 2 partitions that could be generated by

input : X, a numeric data matrix of size $n \times m$
$n_{\text{min}}^{\text{cluster}}$, minimum number of rows for a cluster
$r_{\text{min}}^{\text{mdl}}$, minimum ratio of MDL reduction for increasing the number of cluster
output: Clusters of n rows in a list of n ⟨row index, cluster index⟩ pairs

```
1  // Phase 1: divide n rows into sub-clusters
2  // build HAC tree with linear optimal leaf ordering
3  hacTree ← HAClinearOptLeafOrd(X)
4  // divide the tree leaves into sub-clusters with less than n_min^cluster rows
5  subClusters ← DivideHACTree (hacTree, n_min^cluster)
6  // Phase 2: iteratively merge sub-clusters into clusters
7  // Merge beginning and end sub-clusters to form two minimum sub-clusters
8  // with more than n_min^cluster rows
9  MergeSubCls (subClusters, start = 1, minSize =n_min^cluster)
10 n_sub-clusters ← Length (subClusters)
11 MergeSubCls (subClusters, start =n_sub-clusters, step = −1, minSize =n_min^cluster)
12 i ← 2
13 while i ≤ n_sub-clusters do
14     left ← subClusters [i − 1]
15     this ← subClusters [i]
16     // merge this with left or right until this has ≥ n_min^cluster rows
17     while Length (this) < n_min^cluster and i + 1 ≤ n_sub-clusters do
18         right ← subClusters [i + 1]
19         // minimum merged sub-cluster with ≥ n_min^cluster rows starting from i + 1
20         rightMM ← MergeSubCls (subClusters, start = i + 1, minSize =n_min^cluster, inPlace =false)
21         if EncodeMDL (left, this) < EncodeMDL (rightMM, this) then // merge left and this
22             left ← MergeSubCls (subClusters, selected = [i − 1, i])
23             this ← right
24         else // merge this and right
25             this ← MergeSubCls (subClusters, selected = [i, i + 1])
26         end
27     end
28     // decide whether merge left and this or not
29     merged ← Concatenate (left, this)
30     mergedMDL ← MDL (merged)
31     splitMDL ← MDL (left) + MDL (this) + SplitOverheadMDL (left, this)
32     if splitMDL < MaxSplitMDL (mergedMDL, r_min^mdl) then // split
33         i ← i + 1
34     else // merge
35         MergeSubCls (subClusters, selected = [i − 1, i])
36     end
37 end
38 return subClusters
```

Fig 2. The minimum description length iteratively regulated agglomerative clustering (MIRAC) algorithm. MIRAC extends hierarchical agglomerative clustering (HAC) in a divide and conquer manner for scRNA-seq data. Input with raw or dimensionality reduced scRNA-seq data, MIRAC starts with building an HAC tree (Line 1–3), and the tree is then divided into small sub-clusters (Line 4–5), which are further merged iteratively into clusters (Line 9–37). The rationales and detailed procedures are described in the Methods section.

https://doi.org/10.1371/journal.pcbi.1007794.g002

merging adjacent clusters of P^s. The upper bound of $M(P^s)$ is the same as the number of step 2 partitions, which could be reached when $k = n$. The lower bound of $M(P^s)$ is not straightforward, since the merged partitions should have all clusters with $\geq t$ cells.

In order to find a good cell partition P^g in the subset of all possibles, we iteratively inspect each cluster of P^s and merge it with either its left or right adjacent cluster according to the similarity determined by the two-stage MDL scheme described in the following sections. The procedure has the following steps:

1. Merge P^s clusters in the beginning of the optimal ordering until the merged set contains $\geq t$ cells (line 9 Algorithm 1).
2. Similarly, merge P^s clusters at the end of the optimal ordering until the merged set contains $\geq t$ cells (line 11 Algorithm 1).
3. For every cluster P^s_i in the middle, determine the similarity between P^s_i and the cluster on the left and right, and merge P^s_i with the more similar cluster (line 20–26 Algorithm 1). Although the cluster on the left P^s_{i-1} always has $\geq n^{\text{cluster}}_{\min}$ cells, the cluster on the right P^s_{i+1} may have a minimum of one cell, so that the determined similarity between P^s_i and P^s_{i+1} is sensitive to noise. In order to improve the robustness of similarity comparison, instead of determining the similarity between P^s_i and P^s_{i+1}, we determine the similarity between P^s_i and $P^s_{\text{rmm}} = P^s_{i+1} \cup P^s_{i+2} \cup \ldots \cup P^s_{i+m}$ (line 20 Algorithm 1), where $|P^s_{\text{rmm}}| \geq n^{\text{cluster}}_{\min}$, $|P^s_{\text{rmm}}| - P^s_{i+m} < n^{\text{cluster}}_{\min}$, and rmm is the shorthand for right minimax.
4. Once $|P^s_i| \geq n^{\text{cluster}}_{\min}$, determine the similarity between P^s_{i-1} and P^s_i and merge them if their similarity is above a certain threshold, otherwise split them (line 32–36 Algorithm 1). The code length of merged $X_{\text{list}(P^s_{i-1} \cup P^s_i)}$ is

$$L_m = L(X_{\text{list}(P^s_{i-1} \cup P^s_i)}).$$

The code length of divided $X_{\text{list}(P^s_{i-1} \cup P^s_i)}$ is

$$L_d = L(X_{\text{list}(P^s_{i-1} \cup P^s_i)}, \{P^s_{i-1}, P^s_i\}).$$

If $L_d - L_m \geq -r^{\text{mdl}}_{\min}\text{abs}(L_m)$, merge P^s_{i-1} and P^s_i, otherwise split them. This conditional statement generalizes the situations where L_m is either non-negative or negative.

5. Continue inspecting the next cluster in P^s.
6. After inspecting all middle clusters, P^g is the final updated P^s.

We use a standard HAC tree to perform MIRAC when the number of samples is larger than ten thousand, which usually results in decreased but still acceptable performances (Fig 3). Although the optimal leaf ordering of the HAC tree is an important assumption, its computation takes too long when the number of samples is large. For example, it takes more than a week to compute the optimal leaf ordering of a dataset with about 68,000 samples. However, the time complexity of computing a good HAC linear ordering could be significantly reduced by implementing other ordering techniques [28].

Analysis of complexity. The time complexity of MIRAC is (n^4+mn^2), where $n,m \geq 1$. The time complexity of computing the HAC tree with optimal leaf ordering is $O(n^4)$ [24]. The time complexity of finding P^g is $O(mn^2)$, which is briefly analyzed as following.

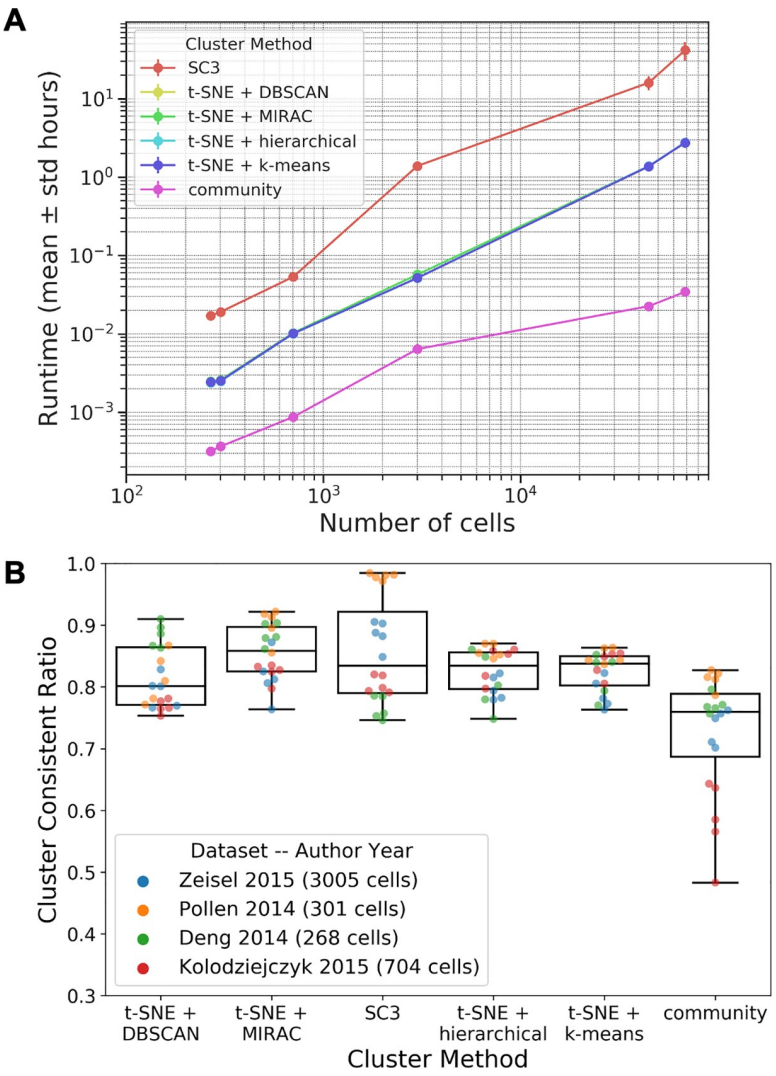

Fig 3. Clustering method benchmarks on experimental datasets. (**A**) Runtimes. (**B**) CCRs on different datasets, with different points of each dataset representing different numbers of clusters. For each dataset, the numbers of clusters are the same across all compared clustering methods.

https://doi.org/10.1371/journal.pcbi.1007794.g003

The time complexity is $O(nm)$ for computing the code length $L(X_{n\times m},P)$. In computing $L(X_{n\times m},P)$ we compute the code lengths of each cluster and their labels. Since it takes $O(n)$ time to compute the code length of n observations with a single feature, computing the code length of all clusters and features individually takes $O(nm)$ time, and computing the code length of n labels take $O(n)$ time.

Similarly, the time complexity is $O(n_1m+n_2m)$ for computing the code length of $X_{n_1\times m}$ encoded by a model fitted with $X_{n_2\times m}$. Fitting a model on $X_{n_2\times m}$ takes $O(n_2m)$ amount of time. Using the model to encode $X_{n_1\times m}$ takes $O(n_1m)$ amount of time.

The time complexity of searching for ^{Pg} is $O(mn^2)$. In the worst-case scenario, $|P^s| = n$, and the inspecting cluster is always merged with the left-hand side cluster in step 4. However, the impact of $n_{\min}^{\text{cluster}}$ and step 3 on the time complexity is not straightforward, so we divide the step 3 code length computation into three parts and analyze the upper bound of them individually. The divided three parts are the left cluster, inspecting cluster, and right minimax cluster.

Because we keep increasing the size of the left cluster, the upper bound of computing the left cluster code length throughout the execution is $O(mn^2)$. The maximum size of the inspecting cluster is $n_{\min}^{\text{cluster}} - 1$, so the upper bound of computing the inspecting cluster code length is also $O(mn^2)$, when every middle singleton is evaluated $n_{\min}^{\text{cluster}} - 1$ times before merging with the left. The maximum size of the right minimax cluster is $2n_{\min}^{\text{cluster}} - 1$, so the upper bound of evaluating the right minimax cluster is also $O(mn^2)$, when every increment of the left cluster takes $n_{\min}^{\text{cluster}}$ times of evaluating the right minimax cluster. Summarizing these three parts, the overall upper bound is therefore $O(mn^2)$.

The space complexity of MIRAC is $O(nm+n^2)$, which could be decomposed into the following three parts. The space complexity of storing the $n \times m$ data matrix X is $O(mn)$. The space complexity of storing the HAC tree with optimal leaf ordering is $O(n)$. The space complexity of storing the pairwise distance matrix is $O(n^2)$.

Extension with community detection. We extended MIRAC with community detection to improve scalability. We apply MIRAC on a relatively large number of detected single-cell communities to identify final clusters. We used the Leiden algorithm for community detection on KNN graphs [29]. We also provide a KNN graph construction method that supports approximate nearest neighbor (ANN) search using a Hierarchical Navigable Small World (HNSW) graph [30].

Cluster separating genes identification

We use XGBoost [18], a scalable and sparsity-aware boosted tree system, to identify genes that are able to separate a specific set of clusters. This method is designed for data exploration after applying any one of a number of various statistical approaches that have been developed to identify differentially expressed genes [31–35], in order to quickly identify genes or sets of genes that are able to separate an arbitrary set of clusters for further inspection.

Rather than providing a meticulous p-value for each gene among the compared clusters, we rank the genes by their individual importance on separating the clusters under comparison. The importance of a gene in the trained classifier is the number of times it has been used as an inner node of the decision trees. We use cross validation to train an XGBoost classifier on the compared clusters, and the classifier is essentially a bag of decision trees [18]. In order to alleviate the influences of stochasticity on interpretation, we use bootstrap with feature shuffling to better estimate the importance of genes in separating the compared clusters. The obtained list of important genes could be further explored by inspecting transcription level fold changes and decision tree structures.

Comparing to NSForest [36], a method based on random forest [37,38] to identify a parsimonious set of cluster separating genes from scRNA-seq data, our method identifies all possible cluster separating genes using gradient boosting. Practically, NSForest version 1.3 is distributed on GitHub as a Python script without encapsulation or testing (checked on Oct 11, 2018), but our method is distributed through the Python Package Index, comprehensively tested, and easy to use through the user-friendly API. With regard to scalability, our method uses a scalable implementation of gradient boosting algorithm [18], whereas NSForest uses the implementation of random forest in scikit-learn [38].

K-nearest neighbor methods

A k-nearest neighbor analytical strategy exploits the similarity between data points for classification, regression, and imputation [39,40]. In k-nearest neighbor methods, samples are considered as points in a space with each dimension representing a measured property, which is often referred to as a feature, of the samples. The similarity between samples can be evaluated

with various distance metrics, which is extensively reviewed by Bellet *et al.* [41]. Generally, a distance metric is a function that takes two samples and output a numeric value to represent the distance between the two samples in their feature space. For example, the Euclidean distance metric is the following function,

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

where the $\boldsymbol{p}$ and $\boldsymbol{q}$ are two samples, and q_i and p_i are the data values on their ith dimension. The k-nearest neighbors of a sample are the k number of other samples that have the smallest distances from the sample. The k-nearest neighbors of a sample are informative, due to their similarity, to determine the category of the sample in classification, the relevant continuous property in regression, and the missing values in imputation.

We developed two methods based on the KNN algorithm to facilitate the exploration of scRNA-seq datasets. For a relatively large number of cells profiled in each scRNA-seq experiment, we assume that each one of the non-rare cells is similar to at least k other cells in their transcriptomic profiles. With this assumption, we impute gene dropouts and detect rare transcriptomic profiles. In the scedar implementation, we also support approximate nearest neighbor (ANN) search using Hierarchical Navigable Small World (HNSW) graph [30], which greatly improves the scalability of KNN graph construction.

Impute gene dropouts. In an scRNA-seq experiment, if a truly expressed gene is not detected in a cell, the gene is considered a "dropout", and such events are called gene dropouts [32]. Gene dropouts may be caused by biological and technical reasons [32,42,43]. The rationale behind the possible causes of biological dropouts are related to phenomena such as transcriptional bursting [44,45] and RNA degradation. With regard to technical dropouts, the main concerns are the relatively small number of RNA transcripts of a gene, amplification efficiency, and batch effects [13].

We exploit the transcriptomic profiles of the k-nearest neighbors of a cell to impute the gene dropouts in the cell (Algorithm A in S1 Text). The algorithm could take multiple iterations, so that the dropped-out genes that are expressed in all k-nearest neighbors could be imputed at first, and the ones that are expressed in most but not all of k-nearest neighbors could be imputed in the following iterations.

Detecting rare transcriptomic profiles. We mark transcriptomic profiles as rare if they are distinct from their k-nearest neighbors, according to the pairwise similarity between cells (Algorithm B in S1 Text). The algorithm could take multiple iterations, so that the most distinct transcriptomic profiles could be marked at first and less distinct ones in the following iterations.

This method is provided mainly to facilitate detailed inspection of rare transcriptomic profiles rather than removing outliers from the data. Because rare transcriptomic profiles may have various biological and technical causes, samples and features in a dataset should only be removed after extensive exploratory data analysis and rigorous reasoning with domain specific knowledge. Closely comparing rare transcriptomic profiles with their nearest neighbors may also yield insights into their biological differences, which may further facilitate the identification of rare cell types and states.

Benchmarking

We benchmarked the clustering and KNN performances of scedar on simulated and experimental scRNA-seq datasets. We obtained previously published experimental scRNA-seq datasets (Table 1). We also generated 50 simulated scRNA-seq read count datasets with Splatter

Table 1. Real scRNA-seq datasets for benchmark.

Publication	# cells	# genes	Organism	Tissue	Protocol	Raw Data Type
Deng *et al.* (2014)	268	22,431	Mus Musculus	Embryo	Smart-Seq/Smart-Seq2	Read count
Pollen *et al.* (2014)	301	23,730	Homo sapiens	Dermal, blood, pluripotent and neural	SMARTer	TPM
Kolodziejczyk *et al.* (2015)	704	38,653	Mus Musculus	Embryonic stem cell	SMARTer	Read count
Zeisel *et al.* (2015)	3005	19,972	Mus Musculus	Brain	STRT-Seq UMI	Read count
Macosko *et al.* (2015)	44,808	23,288	Mus Musculus	Retina	Drop-seq	Read count
Zheng *et al.* (2015)	68,579	33,694	Homo sapiens	Peripheral blood mononuclear cell	GemCode	Read count
Han *et al.* (2018)	405,191	39,855	Mus Musculus	Various tissues	Microwell-Seq	Read count
Cao *et al.* (2019)	2,058,652	26,183	Mus Musculus	Embryo	sci-RNA-seq3	Read count

Source: https://hemberg-lab.github.io/scRNA.seq.datasets/. TPM represents Transcripts Per Million in the raw data type column.

https://doi.org/10.1371/journal.pcbi.1007794.t001

[13]. The simulation parameters are estimated by Splatter according to a Drop-seq dataset [15] and a 10x Genomics GemCode dataset [16]. Within each simulated dataset, the cells have 8 clusters taking approximately the following proportions: ⟨0.3, 0.2, 0.15, 0.15, 0.05, 0.05, 0.05, 0.05⟩, with a gene dropout rate around 5%.

We performed all benchmark analyses on a high-performance computing cluster, of which the computing resources are strictly managed by Univa Grid Engine. The cluster nodes have CPUs of Intel Xeon E5-2680 v3 or Intel Xeon E7-8880 v3. The memory sizes are either 128GB, 256GB, or 1TB. When scheduling analytical jobs for benchmarking, we make sure that the number of cores and allocated memory are enough for the program.

Clustering. The clustering accuracy and stability of MIRAC were benchmarked together with several other clustering methods on experimental scRNA-seq datasets listed in **Table 1**.

The following clustering methods are directly applied on the original data without preprocessing.

- MIRAC on 2D t-SNE projection.
- Single-cell consensus clustering (SC3) version 1.7.7 [46]. SC3 is selected for comparison because it was extensively compared with other methods on different experimental datasets.
- K-means clustering on 2D t-SNE projection.
- Hierarchical agglomerative clustering on 2D t-SNE projection.
- Density-based spatial clustering of applications with noise (DBSCAN) [47] on 2D t-SNE projection.
- Community clustering on KNN graph constructed using the Euclidean distances of 100 principal components. We used the Leiden algorithm for community detection [29].

Although MIRAC could be directly applied on the expression matrix, dimensionality reduction is able to improve the performance of similarity and density-based clustering methods when the number of features is high [48]. The mathematical influences of the high number of features are briefly described in the previous sections.

Although t-SNE projections are stochastic and influenced by the perplexity parameter, t-SNE is proved to be able to recover well-separated clusters [49] and t-SNE has been extensively used as dimensionality reduction method for scRNA-seq data [3,15]. Thus, we chose to use t-SNE for demonstration in this report, but we note that scedar also supports the use of PCA and UMAP for MIRAC clustering, which can be applied just as easily as the t-SNE method.

When benchmarking for accuracy, we cluster the cells in each experimental dataset using the compared clustering methods with a grid of parameters. We use the maximum similarities between the clustering results and the cell types from the publications in order to compare the accuracy of different clustering methods. Although taking the maximum increases the chance of overfitting, it resembles the procedure of clustering analysis in practice.

We also recorded the runtime of clustering methods on different datasets. In Fig 3A, SC3 was performed with 20 cores, and community clustering is performed with 60 cores. Other clustering methods were performed with a single core. When running MIRAC, we did not require the hierarchical tree to be optimal. The community-detection-extended MIRAC clustering method was performed with 25 CPU cores, and other methods were performed with 60 CPU cores (S8 Fig, panel C).

We also benchmarked the influences of t-SNE random states on the clustering results (S1 Fig). The t-SNE embeddings generated with different random states may be distinct from each other, although current mathematical results have shown that clustering using t-SNE embeddings is able to recover well separated clusters [49]. We characterized the stability to random states of clustering methods by running them on each experimental dataset with the same parameters but ten different random states. The similarity of clustering results between different random states are used to compare the stability of different clustering methods. The stability of community clustering is not characterized, because it is based on PCA.

Cluster similarity metrics. We use two cluster similarity metrics, cluster consistent ratio (CCR) and adjusted Rand index (ARI) [50] for measuring the accuracy and stability of clustering methods respectively. When we have a coarse reference partition P_r and a finer clustering partition P_c, the CCR is computed as the ratio of pairs within each cluster of P_c that are also in the same cluster of P_r, with the number of clusters kept the same across compared methods. The ARI is computed with the Python package scikit-learn [38] using the mathematical formula given by Hubert and Arabie [50].

The reference partitions P_r of real datasets are obtained from their original publications (Table A in S1 Text). The clusters in Deng *et al.* dataset [51] are experimentally determined by manual single cell isolation. The clusters in Pollen *et al.* [14] are experimentally determined by the cell line or tissue type of the isolated single cells. The clusters in Kolodziejczyk *et al.* [52] are determined by the embryonic stem cell culturing conditions and batches. The clusters in Zeisel *et al.* [53] are determined by the BackSPIN clustering method [53] and further inspected with domain specific knowledge.

We choose CCR to measure clustering accuracy rather than ARI, because ARI greatly penalizes the split of a large cluster in P_r into multiple smaller ones in P_c. This behavior of ARI could prevent the evaluation of transcriptomic variabilities within a large group of cells in clustering analysis, which is an important goal of scRNA-seq experiments that cannot be easily achieved by bulk RNA-seq experiments. In addition, we also provide a method in scedar to easily merge multiple clusters together, in case the users found the sub-types of a cell type are very similar to each other.

However, we used ARI to measure clustering stability rather than CCR, because the differences between P_r and P_c are completely caused by different random states, hence splitting a cluster in P_r should be penalized.

MDL is not used as a cluster similarity metric, even though MDL is used in MIRAC to guide the process of finding good partitions. MDL is only used to guide and regularize the merging process of local adjacent sub-clusters by evaluating their similarity, but it is not used as an objective to be optimized globally. Using MDL as a benchmarking metric would introduce a bias as among the various methods, MIRAC is the only method using MDL. Moreover,

interpreting relative global MDL differences is not straightforward as it is not only affected by cluster labels, but also transcription levels.

Detection of rare transcriptomic profiles. We visualize real datasets before and after removing rare transcriptomic profiles in t-SNE projection and pairwise distance heatmaps, without quantitative evaluations like receiver operating characteristic (ROC) curve, since rare transcriptomic profiles are not well defined with a large number of genes [48]. Approaches to identify rare data points in low-dimensional spaces ($\leq$3) do not scale well to higher dimensions due to the exponentially decreased density of data points in space and increased instability of distances, which is elaborated in the previous section about mathematical theories on high-dimensional data analysis. KNN rare transcriptomic profile detection was performed with 25 CPU cores, as shown in S8 Fig panel B.

It is important to note that rare transcriptomic profiles are detected to facilitate detailed inspection rather than the removal of them from the data. The visualizations are only used to illustrate the capability of the KNN method for detecting rare transcriptomic profiles.

Gene dropout imputation. We simulate gene dropouts using Splatter [13] to obtain a dropout rate around 5%. Then, we benchmark the performance of imputing gene dropout as two parts: detection and smoothing. On simulated data, because the true gene dropouts are known, we use a ROC curve and mean squared errors (MSEs) to characterize the performance of gene dropout detection and smoothing respectively. In Fig 4A, S7 Fig panel A, and S8 Fig panel A, all imputation methods were executed with a single core. On real data, we visualize the cells in 2D t-SNE space before and after imputation. The compared methods are KNN gene dropout imputation (KNNGDI) version 0.1.5, SAVER version 1.0.0 [54], MAGIC version 1.1.0 [55], and scImpute version 0.0.6 [56].

Method speed-up by parallel processing. We characterized the analysis speed-up by parallel processing on different datasets (S9 Fig). KNN dropout imputation is not evaluated on larger datasets, because the speed limiting step has not yet been parallelized. In addition to parallelizing the computation in the implemented methods, we also provide an easy-to-use utility function (scedar.utils.parmap) to parallelly run analytical methods with different parameters, which would significantly facilitate exploratory parameter search.

Results

Basic workflow of scedar

We illustrate the basic workflow of using scedar for scRNA-seq exploratory data analysis with the dataset published by Zeisel *et al.* [53] (Fig 1). The dataset contains the RNA unique molecule identifier (UMI) counts of 19,972 genes in 3005 cells from the mouse brain. We selected this dataset for demonstration because it could be clearly visualized in small graphs, although our package is capable of analyzing much larger datasets with over 2 million cells (S2 Fig, S3 Fig, S4 Fig and S5 Fig).

In Fig 1, each box represents a data analysis step, and they are consecutively executed according to the arrow. The purpose and runtime are listed in the upper ribbon. The code that is essential to the step is listed in the box, and their results are also shown.

The preparation step imports required packages and loads the data. The class SampleDistanceMatrix is one of the core data structures in the package that is used to store the read counts and pairwise distances of an scRNA-seq dataset. Because the pairwise distance computation is delayed until necessary, i.e. lazily loaded, this step only takes 12 seconds. We use cosine distance rather than correlation distance to greatly speed up the computation, since we implemented the computation procedure of pairwise cosine distances completely with NumPy linear algebra operations with OpenBLAS backend [57,58].

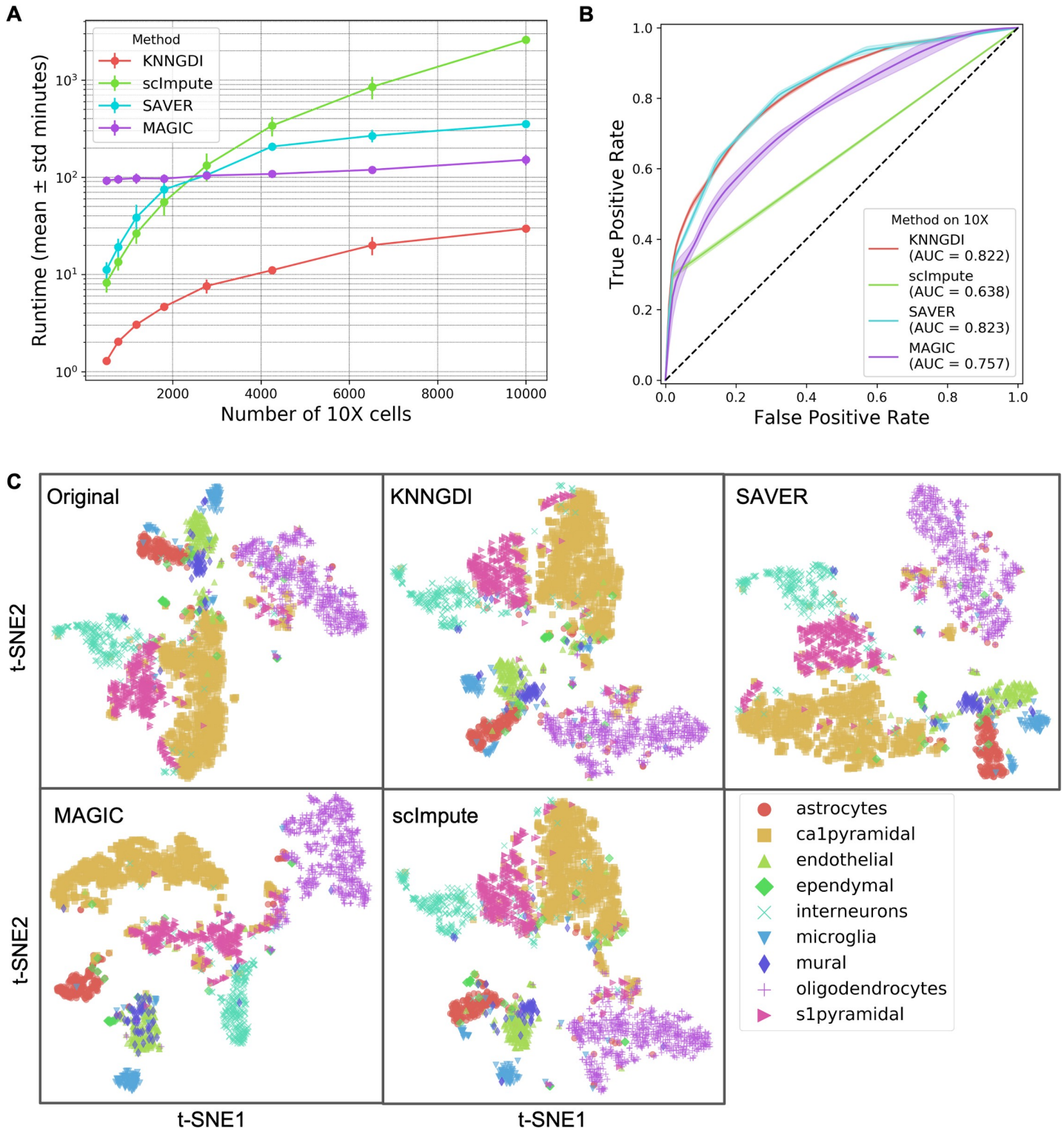

Fig 4. Gene dropout imputation method benchmarks. (**A**) Runtimes on 40 simulated 10x Genomics datasets. (**B**) ROC curves (± standard deviation) of dropout detection on the simulated 10x Genomics datasets. (**C**) t-SNE scatter plots of the Zeisel *et al.* [53] dataset after gene dropout imputations.

https://doi.org/10.1371/journal.pcbi.1007794.g004

The t-Distributed Stochastic Neighbor Embedding (t-SNE) scatter plot and KNN graph are used to explore the dataset. The cell type labels published by Zeisel *et al.* [53] are truncated to fit in the space. We also provide methods to visualize arbitrary statistics, e.g. number of expressed genes, of individual cells as color gradient. The layouts of cells in t-SNE and KNN graph are similar to each other. Although KNN graph is faster than t-SNE, the runtime for t-SNE could be reduced by optimizing its parameters, e.g. lowering the number of iterations.

The MIRAC step clusters the cells and visualizes them with t-SNE scatter plot and pairwise distance matrix heatmap. The heatmap generation procedure in scedar is optimized for large-scale datasets, which is able to generate a heatmap with tens of thousands of columns and rows in a few minutes. Users could also generate heatmaps for the read count matrix to directly inspect the sparsity of datasets (all panels B in S2 Fig, S3 Fig, S4 Fig and S5 Fig).

The last step identifies cluster separating genes with XGBoost [18]. Users could choose an arbitrary set of clusters to compare, and the genes are ranked by their importance in separating the clusters. Then, the read counts of a gene across clustered labels could easily be visualized by t-SNE scatter plot.

Performance of MIRAC clustering

We benchmarked several clustering methods on the datasets listed in Table 1 (Fig 3). Each dataset is clustered multiple times with each clustering method to obtain different numbers of clusters (Table A in S1 Text).

The t-SNE based clustering methods are faster than SC3 (Fig 3A). Also, the t-SNE based clustering methods have similar runtimes (Fig 3A), since the time limiting step is the computation of t-SNE projection, when optimal hierarchical clustering ordering is not required in MIRAC. When the optimal ordering is required, the time limiting step is the computation of optimal ordering (Fig 1). With regard to practical usage, we recommend the users to explore different parameters of MIRAC without optimal ordering. Once appropriate parameters have been identified, users can perform MIRAC with optimal ordering. However, for datasets with more than 10,000 cells, we recommend that users perform MIRAC without optimal ordering (S2 Fig panel C and S3 Fig panel C). For datasets with more than 100,000 cells, we recommend that users perform community-detection-extended MIRAC without optimal ordering (S8C Fig), which takes for instance 2885.5 ± 176.0 (mean ± standard deviation) seconds on the mouse organogenesis cell atlas (MOCA) dataset with 2,058,652 single cells using 25 CPU cores (S8C Fig) [59]. We also performed clustering with the community clustering methods implemented in scedar, Seurat [8] and Scanpy [9] on relatively large scRNA-seq datasets with more than 10,000 cells (S8C Fig). The runtime of community clustering methods Seurat and Scanpy is comparable to community clustering and community-detection-extended MIRAC (S8C Fig), except that Scanpy and Seurat were not able to cluster the MOCA dataset that contain 2,058,652 single cells on a server with 1TB memory due to a mandatory conversion of the sparse read count matrix into dense matrix.

The cluster consistent ratios (CCRs) of t-SNE based clustering methods are comparable to SC3 (Fig 3B). The PCA based community clustering results showed relatively lower CCRs than other clustering methods (Fig 3B), which might due to the non-optimal representation of similarities between high-dimensional single cell read counts in the linear PCA space, comparing to t-SNE embeddings that are optimized to capture the similarities between high-dimensional single cell read counts. The representative MIRAC clustering results of Zheng *et al.* [16] and Macosko *et al.* [15] datasets are visualized with t-SNE scatter plots and pairwise distance matrix heatmaps (S2 Fig panel C and S3 Fig panel C). For smaller datasets, the representative MIRAC results are shown (S6 Fig). Although t-SNE projections obtained with different

random states are distinct from each other, the consistency of clustering results is comparable to SC3 (S1 Fig).

Performance of imputing gene dropouts

We benchmarked several gene dropout imputation methods on the simulated 10x Genomics (Fig 4) and Drop-seq (S7 Fig) datasets. K-nearest neighbor gene dropout imputation (KNNGDI) is faster than other compared methods on relatively small datasets (Fig 4A) and is capable of handling scRNA-seq datasets with over 10,000 cells (S8 Fig panel A). The runtime of KNNDGI on 2,058,652 cells is 71.56 ± 1.71 (mean ± standard deviation) hours.

The performance of KNNGDI on detecting gene dropouts is comparable to SAVER and better than scImpute and MAGIC. The ROC curve of scImpute sharply turns around TPR = 0.25 (Fig 4B and S7 Fig panel B) because its threshold parameters, determining whether a zero entry is a dropout or not, are not sensitive enough to achieve any higher TPRs. Although the AUCs of KNNGDI and SAVER are higher than scImpute and MAGIC, they all have comparable performances when FPRs are lower than 0.05 (Fig 4B), except that MAGIC has worse performance on the simulated Drop-seq datasets that have higher sparsity than the Zeisel *et al.* [53] dataset (S7 Fig panel B).

The MSEs of KNNGDI on correcting gene dropouts are comparable to SAVER and smaller than scImpute and MAGIC (S7 Fig panel C and S7 Fig panel D). However, these methods all have MSEs many times higher than the MSEs of the observed counts to the true counts, which implies that these methods all introduced many times more reads than the true dropout reads. Despite different levels of MSEs, the t-SNE embeddings of the imputed read counts are consistent with the t-SNE embeddings of original read counts (Fig 4C, and all panels E in S2 Fig, S3 Fig, S4 Fig and S5 Fig).

Performance of detecting rare transcriptomic profiles

We detected rare transcriptomic profiles in datasets listed in Table 1 with the KNN method (Fig 5, and all panels F in S2 Fig, S3 Fig, S4 Fig, S5 Fig and S6 Fig) Although the detection method has many limitations, rare transcriptomic profiles tend to be points on the t-SNE scatter plots either far away from the majority of their same types or along the edges of a group of

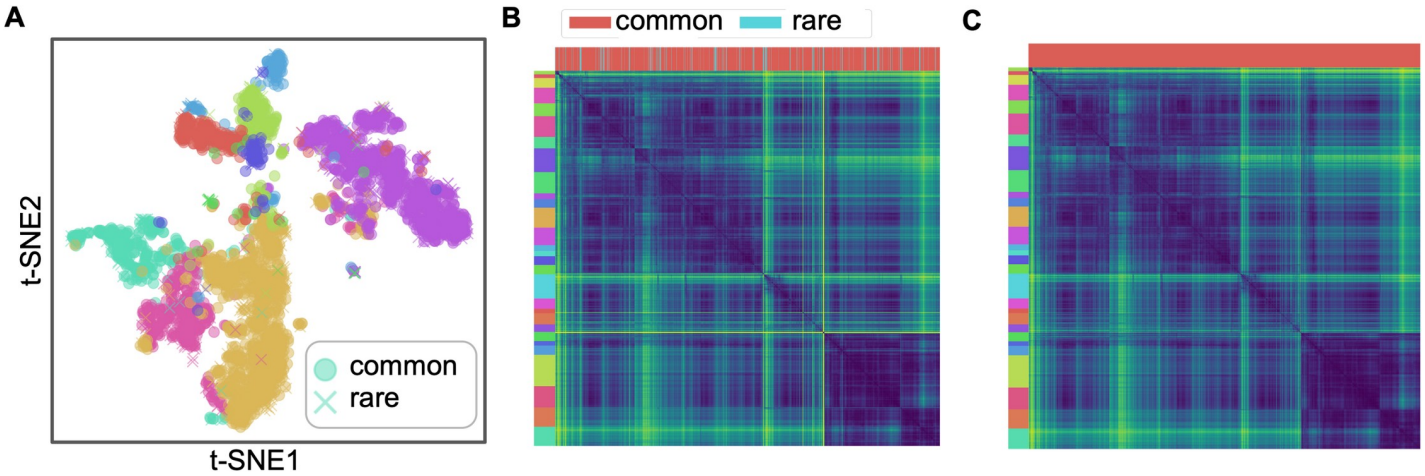

Fig 5. KNN rare transcriptomic profile detection on the Zeisel *et al.* [53] dataset. (**A**) t-SNE scatter plot with colors labeling cell types and markers labeling common or rare transcriptomic profiles. 9.3% cells are marked as rare. (**B**) Pairwise cosine distance heatmap with left strip as MIRAC labels and upper strip as common or rare transcriptomic profiles labels. (**C**) Pairwise cosine distance heatmap with rare transcriptomic profiles removed.

https://doi.org/10.1371/journal.pcbi.1007794.g005

agglomerated points. On the pairwise distance matrix heatmap, rare transcriptomic profiles tend to be small chunks that are distinct from their neighbors, and the heatmap becomes smoother after removing the rare transcriptomic profiles. The detected rare transcriptomic profiles could be further inspected as potential rare cell types and states by comparing with their nearest neighbors. The KNN rare transcriptomic profile detection method is also capable of handling scRNA-seq datasets with over 10,000 cells (S8 Fig panel B), with a runtime of 22.14 ± 0.87 (mean ± standard deviation) minutes on 2,058,652 cells.

Identification of cluster separating genes

We used scedar to identify the genes distinguishing the MIRAC cluster 1, 15, and 22 of the Zeisel *et al.* [53] dataset (Fig 6 and Table B in S1 Text). In the original publication, the upper to lower MIRAC clusters labeled 22, 1, and 15 in the t-SNE scatter plot are assigned to microglia, endothelial cells, and astrocytes respectively (Fig 4C). We choose these three clusters to inspect one of the discrepancies between cell types and MIRAC clustering results, where the small isolated upper part of the MIRAC cluster 15 is assigned to microglia instead of endothelial cells in the original publication.

The smaller upper isolated part of MIRAC cluster 15 might be a distinct cell sub-type of microglia. Although it expresses a microglia marker gene C1qb (Fig 6B) [60], it does not express Mrc1 or Apoe in the same pattern as the MIRAC cluster 22 (Fig 6B and 6C). According to the transcription levels of the cluster separating genes (Fig 6C), the smaller upper isolated part of MIRAC cluster 15, which is located at the top of the cluster 15 rows, has some genes expressed in the same pattern as the microglia, but it also has some other genes expressed distinctly from the microglia.

Discussion

Comprehensive profiling of the transcriptomes of individual cells within an organism or tissue by scRNA-seq is able to facilitate the systematic study of physiological or pathological states [3,61,62]. Previous scRNA-seq experiments identified novel cell types or states [61], obtained insights into the regulation of differentiation process [3,59,63,64], and inferred molecular mechanisms of tissue functions [53,65,66].

The biological results of scRNA-seq experiments are obtained from extensive data analyses, which could take more time than doing the experiments. As the size of scRNA-seq datasets increases rapidly [1,67], computational methods also need to be scalable by exploiting hardware and software techniques for big data analytics, in order to complete an analysis within a reasonable amount of time.

Scedar is able to facilitate gene dropout imputation, rare transcriptomic profile detection, clustering, and identification of cluster separating genes for scRNA-seq data by exploiting scalable system design patterns and high-performance computing architectures. We parallelized time-consuming computational procedures by multi-processing without excessive copies of shared data in memory. We also decompose the whole analytical procedure into multiple steps, so that certain steps could be specifically optimized without repeating others. In addition, intermediate results, such as pairwise distances and t-SNE projections, are lazy loaded and cached in a unified data structure to speed up analytical routines, prevent repeated computations, and alleviate the burden on users to keep track of all intermediate results.

Comparing to other computational tools that were developed or updated for large scale scRNA-seq data analysis, like PAGODA2 [68], Seurat v2.0 [8], and Scanpy [9], scedar distinguishes itself with an additional research focus on developing new analytical methods, including those based on machine learning. In scedar, we adapted KNN, a typical machine learning

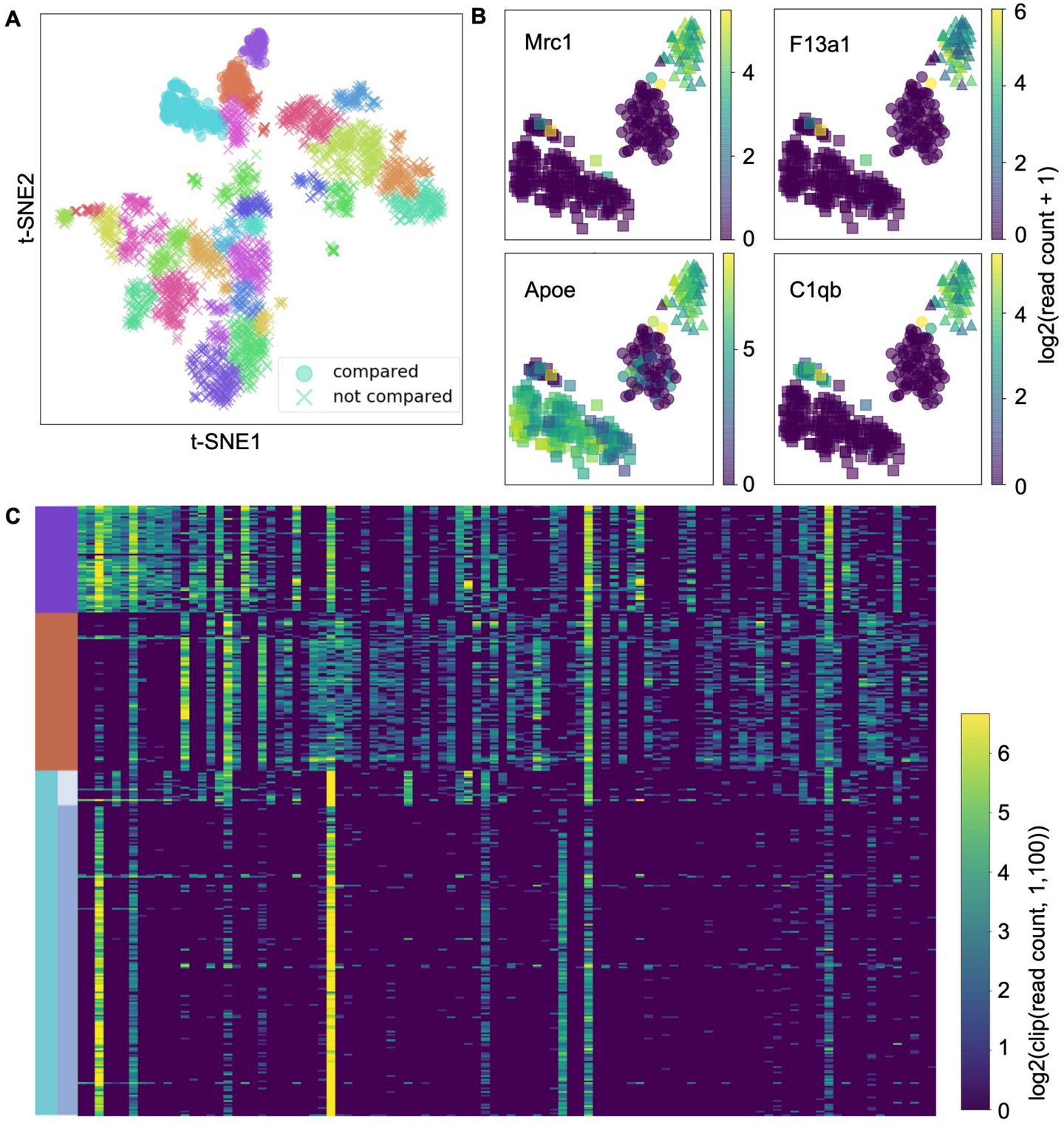

Fig 6. Identified genes separating the MIRAC clusters 1, 15, and 22 of the Zeisel *et al.* [53] dataset. (**A**) t-SNE scatter plot with color as MIRAC cluster labels and marker shape as compared or not compared. (**B**) t-SNE scatter plots of the compared clusters with color as $\log_2$(read count + 1) of the corresponding gene and marker shape as MIRAC clusters. (**C**) Transcription level heatmap of the top 100 important cluster separating genes in the compared cells, with rows as cells ordered by cluster labels and columns as genes ordered by importance. The color gradient is $\log_2$(clip(read count, 1, 100)), where the clip(read count, 1, 100) function changes any read count below 1 to 1 and above 100 to 100, in order to better compare genes at different transcription levels.

https://doi.org/10.1371/journal.pcbi.1007794.g006

algorithm, to impute gene dropouts and detect rare transcriptomic profiles. MDL principle, an important concept in computational learning, is applied to cluster single cells [25]. A scalable and sparsity-aware gradient boosted tree system XGBoost [18], which implements a typical machine learning algorithm, is used to identify genes that are able to distinguish different clusters. In addition, these machine learning methods are able to exploit modern high-performance computing architecture, which improves the scalability of the package.

In scedar, we developed a clustering algorithm, MIRAC, for scRNA-seq data. MIRAC clusters observations in three steps: 1) build a tree by hierarchical clustering, 2) divide the tree into sub-trees, and 3) merge similar sub-trees into individual clusters. This clustering strategy adapts the basic ideas of BIRCH [69] and BackSPIN [53]. Instead of building a balanced tree structure, like the clustering feature tree in BIRCH for further partitioning, MIRAC divides the tree structure built by hierarchical clustering optionally in a balanced manner (S10 Fig and S1 Text section 1.4), which simplifies the clustering procedure. In BackSPIN, a sorted pairwise correlation matrix is recursively bi-partitioned into clusters according to criteria based on the normalized sums of correlation coefficients. In contrast, MIRAC iteratively merges the relatively small sub-clusters according to criteria based on the MDL principle, which increases the robustness for determining whether two groups of observations should be put in the same cluster or not. Especially, when the number of observations is large within a group, finding an optimal bi-partition is not straightforward, since there may be multiple distinct sub-groups. Although the performance of MIRAC under certain metrics is comparable to other clustering algorithms, it is able to provide distinct clusters that are sensitive to local structures, which could be used as alternative perspectives to interpret the source of heterogeneity within the dataset.

There are still many possible improvements on scedar. To improve the scalability of MIRAC, we could provide more efficient methods to obtain the optimal leaf ordering using linear embedding techniques [28]. To improve the scalability of the backend data structure, we could extend it with distributed analytic systems such as Apache Spark [10]. To visualize the differences between different clusters, we could plot the fold changes of gene transcription levels on the pathway maps in the KEGG (Kyoto Encyclopedia of Genes and Genomes) database [70].

Scedar is at its core developed as an open-source, community-extensible package for exploratory data analysis, with strong testing and code development standards (Table C in S1 Text). Scedar will continue to be developed in an open-source environment that can offer an analytical platform for developing novel algorithms, support state-of-the-art software quality control standards, and provide easy-to-use programming interfaces for exploratory data research.

Code availability

The scedar package is distributed through Pypi https://pypi.org/project/scedar

The code and other resources are available in Github at https://github.com/logstar/scedar

Supporting information

S1 Text. Supplementary text. Algorithm A, Algorithm B and Table A, Table B, and Table C. Includes discussion of scedar package development, minimum description length method, two-stage coding scheme for clustered scRNA-seq, mathematical theories on high-dimensional data analysis including distances between points in high dimensional space and the Johnson-Lindenstrauss lemma. Discussion of the skewed root division of a hierarchical agglomerative clustering tree.
(DOCX)

S1 Fig. Stability of clustering methods on experimental dataset. Similarity between clustering results generated with different random states but the same parameters, quantified by adjusted rand index (ARI) [23].
(TIF)

S2 Fig. Scedar analysis of the scRNA-seq dataset containing 44,808 mouse retina cells generated by Drop-seq platform published by Macosko *et al.* [24]. (**A**) t-SNE scatter plot with cell type labels. (**B**) Read count matrix heatmap with rows as cells, columns as genes, and black color as $\geq$ 1reads. (**C**) t-SNE scatter plot with MIRAC labels. (**D**) Pairwise cosine distance heatmap with left strip as MIRAC labels and upper strip as cell type labels. (**E**) t-SNE scatter plot after KNN gene dropout imputation with cell type labels. (**F**) pairwise cosine distance heatmap with left strip as MIRAC labels and upper strip as common or rare transcriptomic profile labels. (**G**) pairwise cosine distance heatmap with rare transcriptomic profiles removed.
(TIF)

S3 Fig. Scedar analysis of the scRNA-seq dataset containing 68,579 human peripheral blood mononuclear cells generated by 10x genomics GemCode platform published by Zheng *et al.* [25]. (**A**) t-SNE scatter plot with cell type labels. (**B**) Read count matrix heatmap with rows as cells, columns as genes, and black color as $\geq$ 1 reads. (**C**) t-SNE scatter plot with MIRAC labels. (**D**) Pairwise cosine distance heatmap with left strip as MIRAC labels and upper strip as cell type labels. (**E**) t-SNE scatter plot after KNN gene dropout imputation with cell type labels. (**F**) Pairwise cosine distance heatmap with left strip as MIRAC labels and upper strip as common or rare transcriptomic profile labels. (**G**) Pairwise cosine distance heatmap with rare transcriptomic profiles removed.
(TIF)

S4 Fig. Scedar analysis of the scRNA-seq dataset containing 405,191 mouse cells from multiple tissues generated by Microwell-seq platform published by *Han et al.* [26]. (**A**) UMAP scatter plot with cell type labels. (**B**) Read count matrix heatmap with rows as cells, columns as genes, and black color as $\geq$ 1 reads. (**C**) UMAP scatter plot with MIRAC labels. (**D**) Pairwise cosine distance heatmap of downsampled 7449 cells with left strip as MIRAC labels and upper strip as cell type labels. (**E**) UMAP scatter plot after KNN gene dropout imputation with cell type labels. (**F**) Pairwise cosine distance heatmap of downsampled 7449 cells with left strip as MIRAC labels and upper strip as common or rare transcriptomic profile labels. (**G**) Pairwise cosine distance heatmap of downsampled 7449 cells with rare transcriptomic profiles removed.
(TIF)

S5 Fig. Scedar analysis of the scRNA-seq dataset containing 2,058,652 mouse cells from multiple tissues generated by sci-RNA-seq3 platform published by *Cao et al.* [27]. (**A**) UMAP scatter plot with cell type labels. (**B**) Read count matrix heatmap of downsampled 20,403 cells with rows as cells, columns as genes, and black color as $\geq$ 1 reads. (**C**) UMAP scatter plot with MIRAC labels. (**D**) Pairwise cosine distance heatmap of downsampled 20,118 cells with left strip as MIRAC labels and upper strip as cell type labels. (**E**) UMAP scatter plot after KNN gene dropout imputation with cell type labels. (**F**) Pairwise cosine distance heatmap of downsampled 20,118 cells with left strip as MIRAC labels and upper strip as common or rare transcriptomic profile labels. (**G**) Pairwise cosine distance heatmap of downsampled 20,118 cells with rare transcriptomic profiles removed.
(TIF)

S6 Fig. MIRAC and KNN rare transcriptomic profile detection results of experimental datasets. The sub-figures A, B, and C represent the results of scRNA-seq datasets published by Pollen *et al.* [28], Deng *et al.* [29], and Kolodziejczyk *et al.* [30] respectively. Within each sub-figure, the plots are 1) t-SNE scatter plot with cell type labels, 2) t-SNE scatter plot with MIRAC cluster labels, 3) pairwise cosine distance heatmap with left strip as MIRAC labels and upper strip as cell type labels, 4) t-SNE scatter plot with common or rare rare transcriptomic profile labels, 5) pairwise cosine distance heatmap with left strip as MIRAC labels and upper strip as common or rare transcriptomic profile labels, 6) pairwise cosine distance heatmap with rare transcriptomic profiles removed.
(TIF)

S7 Fig. Benchmark results of gene dropout imputation methods. (**A**) Runtimes on 40 simulated Drop-seq datasets. (**B**) ROC curves (± standard deviation) of dropout detection on simulated Drop-seq datasets. (**C**) and (**D**) are mean squared error (MSE) ratios of different methods on simulated Drop-seq and 10x Genomics datasets respectively, where the MSE ratio is computed as the MSE of corrected read counts / MSE of true read counts.
(TIF)

S8 Fig. Runtimes of analytical methods implemented in scedar. (A) KNN gene dropout imputation. **(B)** KNN rare transcriptomic profile detection. **(C)** Community clustering, community extended MIRAC clustering, Seurat clustering, and Scanpy clustering. These methods were all performed on experimentally generated scRNA-seq datasets with 3005, 44808, 68579, 405191, and 2058652 cells [24–27,31], except that Scanpy and Seurat were not able to cluster the mouse organogenesis cell atlas (MOCA) dataset that contain 2,058,652 single cells on a server with 1TB memory due to a mandatory conversion of the sparse read count matrix into dense matrix.
(TIF)

S9 Fig. Runtimes of analytical methods implemented in scedar with different number of CPU cores for parallel computation. (A) All implemented methods performed on an scRNA-seq dataset with 3005 single cells [31]. **(B)** KNN rare transcriptomic profile detection, **(C)** community clustering, and **(D)** community extended MIRAC clustering performed on two scRNA-seq datasets with 44,808 and 68,579 single cells [24,25].
(TIF)

S10 Fig. Skewed division of hierarchical agglomerative clustering tree. Tree leaves are samples, which are marked by upper case letters. Tree inner nodes are agglomerated samples by arbitrary linkage, which are marked by number. The triangle under inner node 0 represents an arbitrary valid subtree with $\geq n_{\min}^{\text{cluster}}$ leaves. The root division procedure divides a tree into left and right subtrees of the root node. The skewing procedure creates a minimum subtree of the root with $\geq n_{\min}^{\text{cluster}}$ leaves, where $n_{\min}^{\text{cluster}} = 3$ in this specific case.
(TIF)

Author Contributions

Conceptualization: Yuanchao Zhang, Deanne M. Taylor.

Data curation: Yuanchao Zhang.

Formal analysis: Yuanchao Zhang.

Funding acquisition: Deanne M. Taylor.

Investigation: Deanne M. Taylor.

Methodology: Yuanchao Zhang, Deanne M. Taylor.

Project administration: Deanne M. Taylor.

Resources: Deanne M. Taylor.

Software: Yuanchao Zhang, Deanne M. Taylor.

Supervision: Deanne M. Taylor.

Validation: Yuanchao Zhang, Man S. Kim, Erin R. Reichenberger, Ben Stear.

Visualization: Yuanchao Zhang.

Writing – original draft: Yuanchao Zhang, Deanne M. Taylor.

Writing – review & editing: Yuanchao Zhang, Man S. Kim, Deanne M. Taylor.

References

1. Svensson V, Vento-Tormo R, Teichmann SA. Exponential scaling of single-cell RNA-seq in the past decade. Nat Protoc. 2018; 13: 599–604. https://doi.org/10.1038/nprot.2017.149 PMID: 29494575
2. Filbin MG, Tirosh I, Hovestadt V, Shaw ML, Escalante LE, Mathewson ND, et al. Developmental and oncogenic programs in H3K27M gliomas dissected by single-cell RNA-seq. Science. 2018; 360: 331–335. https://doi.org/10.1126/science.aao4750 PMID: 29674595
3. Cao J, Packer JS, Ramani V, Cusanovich DA, Huynh C, Daza R, et al. Comprehensive single-cell transcriptional profiling of a multicellular organism. Science. 2017; 357: 661–667. https://doi.org/10.1126/science.aam8940 PMID: 28818938
4. Regev A, Teichmann SA, Lander ES, Amit I, Benoist C, Birney E, et al. The Human Cell Atlas. Elife. 2017; 6. https://doi.org/10.7554/eLife.27041 PMID: 29206104
5. Kiselev VY, Andrews TS, Hemberg M. Challenges in unsupervised clustering of single-cell RNA-seq data. Nat Rev Genet. 2019. https://doi.org/10.1038/s41576-018-0088-9 PMID: 30617341
6. Dueck HR, Ai R, Camarena A, Ding B, Dominguez R, Evgrafov OV, et al. Assessing characteristics of RNA amplification methods for single cell RNA sequencing. BMC Genomics. 2016; 17: 377. https://doi.org/10.1186/s12864-016-2735-x
7. Ziegenhain C, Vieth B, Parekh S, Reinius B, Guillaumet-Adkins A, Smets M, et al. Comparative Analysis of Single-Cell RNA Sequencing Methods. Mol Cell. 2017; 65: 631–643.e4. https://doi.org/10.1016/j.molcel.2017.01.023 PMID: 28212749
8. Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol. 2018. https://doi.org/10.1038/nbt.4096 PMID: 29608179
9. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. 2018; 19: 15. https://doi.org/10.1186/s13059-017-1382-0 PMID: 29409532
10. Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, et al. Apache Spark: A Unified Engine for Big Data Processing. Commun ACM. 2016; 59: 56–65.
11. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM. 2008; 51: 107–113.
12. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. Tensorflow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). usenix.org; 2016. pp. 265–283.
13. Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. Genome Biol. 2017; 18: 174. https://doi.org/10.1186/s13059-017-1305-0 PMID: 28899397
14. Pollen AA, Nowakowski TJ, Shuga J, Wang X, Leyrat AA, Lui JH, et al. Low-coverage single-cell mRNA sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex. Nat Biotechnol. 2014; 32: 1053–1058. https://doi.org/10.1038/nbt.2967 PMID: 25086649
15. Macosko EZ, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, et al. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. Cell. 2015; 161: 1202–1214. https://doi.org/10.1016/j.cell.2015.05.002 PMID: 26000488

16. Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun. 2017; 8: 14049. https://doi.org/10.1038/ncomms14049 PMID: 28091601

17. Zappia L, Phipson B, Oshlack A. Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. PLoS Comput Biol. 2018; 14: e1006245. https://doi.org/10.1371/journal.pcbi.1006245 PMID: 29939984

18. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '16. New York, New York, USA: ACM Press; 2016. pp. 785–794.

19. Maaten L van der, Hinton G. Visualizing Data using t-SNE. J Mach Learn Res. 2008; 9: 2579–2605.

20. McInnes L, Healy J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv [stat.ML]. 2018. Available: http://arxiv.org/abs/1802.03426

21. Jacomy M, Venturini T, Heymann S, Bastian M. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. PLoS One. 2014; 9: e98679. https://doi.org/10.1371/journal.pone.0098679 PMID: 24914678

22. Hicks SC, Irizarry RA. quantro: a data-driven approach to guide the choice of an appropriate normalization method. Genome Biol. 2015; 16: 117. https://doi.org/10.1186/s13059-015-0679-0 PMID: 26040460

23. Müllner D. Modern hierarchical, agglomerative clustering algorithms. arXiv [stat.ML]. 2011. Available: http://arxiv.org/abs/1109.2378

24. Bar-Joseph Z, Gifford DK, Jaakkola TS. Fast optimal leaf ordering for hierarchical clustering. Bioinformatics. 2001; 17 Suppl 1: S22–9.

25. Hansen MH, Yu B. Model Selection and the Principle of Minimum Description Length. J Am Stat Assoc. 2001; 96: 746–774.

26. Wilf HS. generatingfunctionology. AK Peters/CRC Press; 2005.

27. Abramson M. Restricted combinations and compositions. Fibonacci Quart. 1976; 14: 439–452.

28. Aydin K, Bateni M, Mirrokni V. Distributed Balanced Partitioning via Linear Embedding. Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. New York, NY, USA: ACM; 2016. pp. 387–396.

29. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep. 2019; 9: 5233. https://doi.org/10.1038/s41598-019-41695-z PMID: 30914743

30. Malkov YA, Yashunin DA. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv [cs.DS]. 2016. Available: http://arxiv.org/abs/1603.09320

31. Vallejos CA, Marioni JC, Richardson S. BASiCS: Bayesian Analysis of Single-Cell Sequencing Data. PLoS Comput Biol. 2015; 11: e1004333. https://doi.org/10.1371/journal.pcbi.1004333 PMID: 26107944

32. Kharchenko PV, Silberstein L, Scadden DT. Bayesian approach to single-cell differential expression analysis. Nat Methods. 2014; 11: 740–742. https://doi.org/10.1038/nmeth.2967 PMID: 24836921

33. Qiu X, Hill A, Packer J, Lin D, Ma Y-A, Trapnell C. Single-cell mRNA quantification and differential analysis with Census. Nat Methods. 2017; 14: 309–315. https://doi.org/10.1038/nmeth.4150 PMID: 28114287

34. Soneson C, Robinson MD. Bias, robustness and scalability in single-cell differential expression analysis. Nat Methods. 2018; 15: 255–261. https://doi.org/10.1038/nmeth.4612 PMID: 29481549

35. Korthauer KD, Chu L-F, Newton MA, Li Y, Thomson J, Stewart R, et al. A statistical approach for identifying differential distributions in single-cell RNA-seq experiments. Genome Biol. 2016; 17: 222. https://doi.org/10.1186/s13059-016-1077-y PMID: 27782827

36. Aevermann BD, Novotny M, Bakken T, Miller JA, Diehl AD, Osumi-Sutherland D, et al. Cell type discovery using single-cell transcriptomics: implications for ontological representation. Hum Mol Genet. 2018; 27: R40–R47. https://doi.org/10.1093/hmg/ddy100 PMID: 29590361

37. Breiman L. Random Forests. Mach Learn. 2001; 45: 5–32.

38. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res. 2011; 12: 2825–2830.

39. Cover T, Hart P. Nearest Neighbor Pattern Classification. IEEE Trans Inf Theory. 2006; 13: 21–27.

40. Bezdek JC, Chuah SK, Leep D. Generalized k-nearest neighbor rules. Fuzzy Sets and Systems. 1986; 18: 237–256.

41. Bellet A, Habrard A, Sebban M. A Survey on Metric Learning for Feature Vectors and Structured Data. arXiv [cs.LG]. 2013. Available: http://arxiv.org/abs/1306.6709

42. Risso D, Perraudeau F, Gribkova S, Dudoit S, Vert J-P. A general and flexible method for signal extraction from single-cell RNA-seq data. Nat Commun. 2018; 9: 284. https://doi.org/10.1038/s41467-017-02554-5 PMID: 29348443

43. Pierson E, Yau C. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. Genome Biol. 2015; 16: 618.

44. Suter DM, Molina N, Gatfield D, Schneider K, Schibler U, Naef F. Mammalian genes are transcribed with widely different bursting kinetics. Science. 2011; 332: 472–474. https://doi.org/10.1126/science.1198817 PMID: 21415320

45. Tantale K, Mueller F, Kozulic-Pirher A, Lesne A, Victor J-M, Robert M-C, et al. A single-molecule view of transcription reveals convoys of RNA polymerases and multi-scale bursting. Nat Commun. 2016; 7: 12248. https://doi.org/10.1038/ncomms12248 PMID: 27461529

46. Kiselev VY, Kirschner K, Schaub MT, Andrews T, Yiu A, Chandra T, et al. SC3: consensus clustering of single-cell RNA-seq data. Nat Methods. 2017; 14: 483–486. https://doi.org/10.1038/nmeth.4236 PMID: 28346451

47. Ester M, Kriegel H-P, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd. 1996. pp. 226–231.

48. Aggarwal CC, Hinneburg A, Keim DA. On the Surprising Behavior of Distance Metrics in High Dimensional Space. Database Theory—ICDT 2001. Springer Berlin Heidelberg; 2001. pp. 420–434.

49. Linderman GC, Steinerberger S. Clustering with t-SNE, Provably. SIAM Journal on Mathematics of Data Science. 2019; 1: 313–332.

50. Hubert L, Arabie P. Comparing partitions. J Classification. 1985; 2: 193–218.

51. Deng Q, Ramsköld D, Reinius B, Sandberg R. Single-cell RNA-seq reveals dynamic, random monoallelic gene expression in mammalian cells. Science. 2014; 343: 193–196. https://doi.org/10.1126/science.1245316 PMID: 24408435

52. Kolodziejczyk AA, Kim JK, Tsang JCH, Ilicic T, Henriksson J, Natarajan KN, et al. Single Cell RNA-Sequencing of Pluripotent States Unlocks Modular Transcriptional Variation. Cell Stem Cell. 2015; 17: 471–485. https://doi.org/10.1016/j.stem.2015.09.011 PMID: 26431182

53. Zeisel A, Muñoz-Manchado AB, Codeluppi S, Lönnerberg P, La Manno G, Juréus A, et al. Brain structure. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. Science. 2015; 347: 1138–1142. https://doi.org/10.1126/science.aaa1934 PMID: 25700174

54. Huang M, Wang J, Torre E, Dueck H, Shaffer S, Bonasio R, et al. SAVER: gene expression recovery for single-cell RNA sequencing. Nat Methods. 2018; 1.

55. van Dijk D, Sharma R, Nainys J, Yim K, Kathail P, Carr AJ, et al. Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. Cell. 2018. https://doi.org/10.1016/j.cell.2018.05.061 PMID: 29961576

56. Li WV, Li JJ. scImpute: Accurate And Robust Imputation For Single Cell RNA-Seq Data. bioRxiv. 2017. p. 141598. https://doi.org/10.1101/141598

57. Xianyi Z, Qian W, Yunquan Z. Model-driven Level 3 BLAS Performance Optimization on Loongson 3A Processor. 2012 IEEE 18th International Conference on Parallel and Distributed Systems. 2012. pp. 684–691.

58. Wang Q, Zhang X, Zhang Y, Yi Q. AUGEM: Automatically generate high performance Dense Linear Algebra kernels on x86 CPUs. SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2013. pp. 1–12.

59. Cao J, Spielmann M, Qiu X, Huang X, Ibrahim DM, Hill AJ, et al. The single-cell transcriptional landscape of mammalian organogenesis. Nature. 2019; 1.

60. Beutner C, Linnartz-Gerlach B, Schmidt SV, Beyer M, Mallmann MR, Staratschek-Jox A, et al. Unique transcriptome signature of mouse microglia. Glia. 2013; 61: 1429–1442. https://doi.org/10.1002/glia.22524 PMID: 23832717

61. Villani A-C, Satija R, Reynolds G, Sarkizova S, Shekhar K, Fletcher J, et al. Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. Science. 2017; 356: eaah4573. https://doi.org/10.1126/science.aah4573 PMID: 28428369

62. Baslan T, Hicks J. Unravelling biology and shifting paradigms in cancer with single-cell sequencing. Nat Rev Cancer. 2017; 17: 557–569. https://doi.org/10.1038/nrc.2017.58 PMID: 28835719

63. Rizvi AH, Camara PG, Kandror EK, Roberts TJ, Schieren I, Maniatis T, et al. Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. Nat Biotechnol. 2017; 35: 551–560. https://doi.org/10.1038/nbt.3854 PMID: 28459448

64. Schiebinger G, Shu J, Tabaka M, Cleary B, Subramanian V, Solomon A, et al. Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming. Cell. 2019; 176: 1517. https://doi.org/10.1016/j.cell.2019.02.026 PMID: 30849376

65. Lake BB, Chen S, Sos BC, Fan J, Kaeser GE, Yung YC, et al. Integrative single-cell analysis of transcriptional and epigenetic states in the human adult brain. Nat Biotechnol. 2018; 36: 70–80. https://doi.org/10.1038/nbt.4038 PMID: 29227469

66. Haber AL, Biton M, Rogel N, Herbst RH, Shekhar K, Smillie C, et al. A single-cell survey of the small intestinal epithelium. Nature. 2017; 551: 333–339. https://doi.org/10.1038/nature24489 PMID: 29144463

67. Rozenblatt-Rosen O, Stubbington MJT, Regev A, Teichmann SA. The Human Cell Atlas: from vision to reality. Nature. 2017; 550: 451–453. https://doi.org/10.1038/550451a PMID: 29072289

68. Fan J, Salathia N, Liu R, Kaeser GE, Yung YC, Herman JL, et al. Characterizing transcriptional heterogeneity through pathway and gene set overdispersion analysis. Nat Methods. 2016; 13: 241–244. https://doi.org/10.1038/nmeth.3734 PMID: 26780092

69. Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases. ACM Sigmod Record. 1996. Available: https://dl.acm.org/citation.cfm?id=233324

70. Kanehisa M, Sato Y, Kawashima M, Furumichi M, Tanabe M. KEGG as a reference resource for gene and protein annotation. Nucleic Acids Res. 2016; 44: D457–62. https://doi.org/10.1093/nar/gkv1070 PMID: 26476454