**scCloud: cloud-based data analysis for large-scale single-cell and single-nucleus genomics**

Bo Li1,2,\*, Joshua Gould1, Yiming Yang1,2, Sirunush Sarkizova1,3,4, Marcin Tabaka1, Orr Ashenberg1, Yanay Rosen5, Michal Slyper1, Monika S Kowalczyk6, Alexandra-Chloe Villani1,2, Timothy Tickle1, Eric Banks1, Anthony Philippakis1, Nir Hacohen1,3, Orit Rozenblatt-Rosen1,\*, Aviv Regev1,7,\*

1Broad Institute of Harvard and MIT, Cambridge, MA, USA

2Center for Immunology and Inflammatory Diseases, Division of Rheumatology, Allergy, and Immunology, Massachusetts General Hospital and Harvard Medical School, Boston, MA 02129, USA

3Center for Cancer Research, Massachusetts General Hospital, Boston, Massachusetts 02114, USA

4Department of Biomedical Informatics, Harvard Medical School, Boston, Massachusetts 02115, USA

5Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, California, 94720, USA

6Celsius Therapeutics, Cambridge, Massachusetts 02141, USA

7Howard Hughes Medical Institute, Koch Institute of Integrative Cancer Research, Department of Biology, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

\*email: [bli28@mgh.harvard.edu](mailto:bli28@mgh.harvard.edu) (BL), [orit@broadinstitute.org](mailto:orit@broadinstitute.org) (ORR) and [aregev@broadinstitute.org](mailto:aregev@broadinstitute.org) (AR)

**Abstract**

Single-cell and single-nucleus genomics revolutionized our ways of understanding complex and heterogeneous tissues and enabled the Human Cell Atlas project and other consortia that aim to understand the human body at the single-cell level1. With the latest technological advances in single-cell genomics2,3, the scientific community is able to produce datasets with millions of cells4,5. The emergence of these large datasets, however, poses a significant challenge for analysis tools, which are not currently fully adapted to handle datasets of this enormous size. To address this pressing challenge, we develop scCloud, a cloud-based data analysis framework that is scalable, cost-effective, able to process a variety of data types and easily accessible to biologists.

Standard single-cell RNA-Seq (scRNA-Seq) data analysis includes three major steps (**Fig. 1a**): sequence read extraction (mkfastq), gene-count matrix generation (count) and mining of biological insights (analysis). scCloud is developed based on Google Cloud Platform and Broad Institute’s Terra service6. It executes the first two steps parallelly across a large number of computer nodes, and executes the last step in a single multi-CPU node using its highly efficient analysis module (**Methods**). As a result, scCloud is much faster than conventional methods. We compared scCloud with the Cell Ranger7 + Seurat8 pipeline using the bone marrow dataset from the human immune cell atlas project9 (**Methods**). scCloud finished the analysis within 15 hours, while the alternative pipeline took almost 5 days to run (**Fig. 1b**). In addition, it only costs around $1.50 on average to analyze one 10x channel using scCloud, compared to the experimental cost of around $2,400 to generate the data (**Fig. 1c, Methods**).

scCloud also supports a more comprehensive set of features **(Fig. 1d)** than other mainstream tools, such as Cell Ranger, Seurat, and SCANPY10. scCloud is the only one of these tools that is cloud based. It is also the only tool to date that supports the analysis (including generating gene-count matrices and mining of biological insights) of both droplet-based scRNA-Seq7 and plate-based scRNA-Seq11(**Methods**). scCloud additionally supports data analysis of CITE-Seq12, which simultaneously measures mRNA expression and the abundance of oligo-tagged surface antibodies at the single-cell level, and pooled CRISPR screens13–17, which simultaneously measure the expression of CRISPR gRNAs and mRNAs at the single-cell level (**Methods**). In addition, scCloud is the only tool that currently supports both cell18 and nucleus19 hashing, important techniques to reduce batch effects and cell/nucleus profiling costs, using a probabilistic demultiplexing algorithm19 (**Methods**). The comprehensive features in scCloud will save users a tremendous amount of time installing different packages for different data types.

Along with the features outlined above, scCloud is designed to be easily accessible to biologists. scCloud users can access computing resources the cloud offers through a simple web-based user interface provided by Terra (**Fig. 1c**). Once the data are analyzed, users can visualize their results instantly using scCloudVis, a serverless web application that enables interactive data visualization and sharing (**Fig. 1d**). Since scCloudVis only downloads to the browser the data that are necessary for visualization (**Methods**), it is scalable to millions of cells. Based on the visualized results, users may want to go back to scCloud to rerun some analyses with adjusted parameters or perform sub-cluster analysis using a proportion of the data (**Fig. 1d**). This iteration can happen several times until users are satisfied with their analyses. Then they can optionally use scSVA20 to generate high quality images for their publications and deposit their data to Single Cell Portal. scCloud is demoed as a featured workspace on Terra (<https://app.terra.bio/#workspaces/fccredits-hassium-crimson-2542/scRNA-seq-cloud>).

scCloud’s analysis module can be run as an independent Python package. This module covers most commonly used scRNA-Seq analysis tasks (**Fig. 2a**, **Methods**). Starting from a gene-count matrix, scCloud filters out low quality data points, selects highly variable genes (HVG) and optionally corrects batch effects. It then performs principal component analysis (PCA) on HVGs to reduce dimension, constructs a k nearest neighbor (kNN) graph on the principal component space, calculates diffusion maps21,22 and applies community detection algorithms to find clusters23,24. scCloud visualizes scRNA-Seq data using either t-SNE25-based or UMAP26-based methods. It can additionally estimate diffusion pseudotimes22 and visualize developmental trajectories using force-directed layout embedding (FLE)27 based algorithms. scCloud can be used to detect cluster-specific markers. It performs differential expression analysis between cells within and outside a cluster and optionally calculates the area under ROC curve (AUROC) values for all genes. It can also train a gradient boosting tree classifier28 on the gene expression matrix to predict cluster labels and outputs genes with high feature importance scores (**Methods**), which provide additional information for detecting cluster-specific markers. Lastly, scCloud annotates clusters with putative cell types based on user-provided legacy signatures. For users’ convenience, scCloud includes cell-type-specific signatures for immune and brain tissues in both human and mouse, which are curated by us (**Methods**).

We have made several algorithm- and implementation-level improvements to boost up the performance of the analysis module. With these improvements, we are able to show that scCloud is orders of magnitude faster than alternatives for running key analyses tasks (**Fig. 2b**, **Methods**). We discuss how each of these improvements works in greater detail below.

Highly variable gene selection is an important feature selection procedure applied to scRNA-Seq data8. We implemented a new HVG selection procedure (**Methods**) in addition to the standard one used by SCANPY10 and Seurat8. The new procedure has an elegant mathematical representation that handles batch effects naturally (**Supplementary Fig. 1a, Methods**). We applied both the new and standard procedures to the bone marrow dataset and compared the selected HVGs with a list of 94 cell-type-specific markers curated for the human immune system based on domain knowledge (**Supplementary Data 1, Methods**). The new procedure selects 3 more markers in its HVG list than the standard procedure (**Fig. 2c**) and includes important T cell markers such as TRAC, CD3D, CD3E and CD4, which are missed by the standard procedure. In addition, the new procedure yields similar cell type annotation (**Supplementary Fig. 1b**) and a developmental trajectory that better separates erythrocyte, B cell and myeloid cell populations (**Supplementary Fig. 1c**).

Batch correction removes technical noise introduced in the process of sample handling, library preparation and sequencing. scCloud implements the classical location and scale (L/S) adjustment method29 for batch correction because we normally have large enough batch sizes (> 25 cells) to estimate mean and variance of each gene separately30. We benchmarked scCloud’s L/S method with Combat30, MNN31 and BBKNN32 that SCANPY offers, and the integration method33 Seurat offers using kBET34 and kSIM acceptance rates (**Fig. 2d**, **Methods**). We observe that there is no single best method that achieves best kBET and kSIM rates. Instead, the five methods evaluated show a trade-off between the two rates. scCloud is the fastest among the five methods (**Supplementary Fig. 2a**) and achieves a good balance between the two acceptance rates (**Supplementary Fig. 2b-g**).

kNN graph construction is a critical step for many downstream analyses, such as community-based clustering, UMAP visualization and diffusion map construction. To make scCloud scalable, we adapt the state-of-the-art approximate nearest neighbor finding algorithm, Hierarchical Navigable Small World (HNSW)35, which is shown to be fastest for high quality approximations36. We compared the HNSW algorithm with the approximate nearest neighbor finding algorithms used by SCANPY and Seurat on the bone marrow dataset (**Methods**). The HNSW algorithm has a near optimal recall (**Supplementary Fig. 3a**) and is orders of magnitude faster than the alternatives (**Supplementary Fig. 3b**).

Diffusion maps are a useful tool for studying developmental trajectories37. scCloud extends the diffusion pseudotime (DPT) algorithm37 by introducing a new family of diffusion maps parameterized by (**Methods**). We show that the DPT method is equivalent to a special case of new diffusion map family with (**Methods**). In addition, we show that the DPT method has a bias towards top diffusion components (**Supplementary Fig. 4a**) and thus reveals less details in the generated developmental trajectory (**Supplementary Fig. 4b**) than the diffusion map scCloud chooses ().

scCloud offers options to cluster single cell data using modularity-based community detection algorithms, such as Louvain38 and Leiden24 (**Methods**). scCloud also proposes new spectral-community-detection algorithms, such as spectral-Louvain and spectral-Leiden, that combine the strengths of both spectral clustering39 and community detection algorithms (**Methods**). We can do spectral clustering easily by applying the k-means algorithm on the calculated diffusion components. Spectral clustering is super-fast, but the clustering results are not satisfiable (**Supplementary Fig. 5a**). The spectral-community-detection algorithms use spectral clustering to aggregate cells into robust groups of cells and then apply community detection algorithms on the aggregated groups (**Methods**). We show that the new algorithms are as scalable as the spectral clustering algorithm and provide clustering results comparable to modularity-based community detection algorithms (**Fig. 2E**, **Supplementary Fig. 5b**).

scCloud can visualize single cell data using either t-SNE25, Fit-SNE40, or UMAP26 (**Methods**). It can also visualize the calculated diffusion map using FLE27 (**Methods**).

**Figure Legends**

**Figure 1. scCloud enables scalable, feature-rich and easily accessible single-cell and single-nucleus genomics data analysis. a.** scCloud data analysis workflow. scCloud takes raw BCL files from sequencers as its input and outputs a variety of analysis results, such as t-SNE plots colored by cell clusters. scCloud has three key computational steps -- *mkfastq*, *count*, and *analysis*. *mkfastq* extracts read sequences as FASTQ files from the BCL files. *count* generates gene-count matrices from the extracted reads. *analysis* pools gene-count matrices from the *count* step and performs a series of analyses, such as highly variable gene selection, PCA, clustering and t-SNE visualization to produce results that enable the discovery of new biology. **b.** scCloud is rapid. We benchmarked scCloud with Cell Ranger + Seurat v3 (running on a 64-CPU server) in terms of total execution time using the 270K bone marrow dataset from the Human Immune Cell Atlas project. **c.** scCloud is cost-effective. We compared the average cost per 10x channel between running scCloud and conducting the wet lab experiment. **d.** scCloud is feature-rich. We compared scCloud with Cell Ranger, Seurat, and SCANPY for features they support. **e.** scCloud enables iterative single-cell data analysis. With raw data, users can use scCloud to produce preliminary analysis results. Then users can instantly visualize the results using scCloudVis. After inspecting the results, users may decide to go back to scCloud to rerun the analysis with adjusted parameters or perform subcluster analysis. This iteration can happen several times. Once the users are satisfied with their results, they can opt to use scSVA to produce publication-quality images and deposit their data to Single Cell Portal.

**Figure 2. scCloud’s analysis module implements key scRNA-Seq analysis tasks efficiently. a.** The analysis module covers most scRNA-Seq analysis tasks. It can filter low quality cells and genes, select highly variable genes, perform batch correction; conduct PCA, build k nearest neighbor graphs, calculates diffusion maps and find clusters based on these graphs; visualize data using t-SNE and UMAP; visualize developmental trajectories using force-directed layouts; perform differential expression analysis, find cluster-specific markers using gradient boosting trees, and annotate putative cell types based on known markers. **b.** scCloud is orders of magnitude faster than Seurat v3 and SCANPY on 9 key analysis tasks. We benchmarked three tools using the full bone marrow data set on a computer server with 28 threads and 256 GB memory. **c.** The new HVG selection procedure selects important T cell markers that the standard procedure misses. We curated a set of 94 human immune cell type markers. The standard procedure (Seurat-style) selects 67 markers as HVGs, while the new procedure (scCloud-style) selects 70 markers. The new procedure successfully selects key T cell markers, such as TRAC, CD3D and CD3E, as well as the key T helper cell marker, CD4 as HVGs, while the standard procedure misses them. **d.** Benchmarking scCloud, ComBat, MNN, BBKNN and Seurat V3 on 34,654 bone marrow cells. Baseline refers to no batch correction. We plot kBET acceptance rate, which assesses how well different batches are mixed, against kSIM acceptance rate, which assesses how well cells with the same cell type are grouped together. An ideal batch corrector should sit in the top right corner. We observe a trade-off between kBET and kSIM acceptance rates among different correction methods. **e**. Spectral community detection algorithm produces clusters of comparable quality in much faster speed. Spectral Leiden algorithm (right) is 32.8x faster than the Leiden algorithm (left) and yields clusters of similar quality (AMI score between the two cluster settings is 0.91). Note that Leiden failed to separate memory B cells from naïve B cells. **f**.

**Supplementary Figure Legends**

**Supplementary Figure 1. Comparison between the new HVG selection procedure (scCloud-style) and the standard HVG selection procedure (Seurat-style). a.** scCloud-style HVG selection in action. We plot variances against means in the log expression space and highlight the LOESS-fitted curve in red. We mark selected HVGs in blue and unselected genes in black. **b.** The two procedures yield comparable results on the bone marrow data set. We show the cell-type-annotated FIt-SNE visualization produced using either Seurat-style (left) or scCloud-style (right) HVG selection procedures. We list adjusted mutual information (AMI) score on top. AMI measures the similarity between two cluster settings. An AMI value close to 1 means highly similar. HSC: hematopoietic stem cell; MSC: Mesenchymal stem cell; cDC: conventional dendritic cell; pDC: plasmacytoid dendritic cell. **c.** scCloud-style HVG selection yields a better developmental trajectory. We visualize the diffusion components for Seurat-style (left) and scCloud-style (right) HVG selection procedures using the force-directed layout algorithm. scCloud-style HVG selection results in a better separation between erythrocyte, B cell and myeloid cell populations.

**Supplementary Figure 2. Benchmarking scCloud, ComBat, MNN, BBKNN and Seurat V3 for batch correction on 34,654 bone marrow cells. a.** Batch correction time for each method. **b.-g.** UMAP visualizations colored by ground truth cell type (left) and 8 donors (right) for **b.** scCloud without batch correction (baseline), **c.** scCloud with L/S adjustment, **d.** ComBat, **e.** MNN, **f.** BBKNN, and **g.** Seurat V3. The 17 ground truth cell types are 1. Naïve T helper cells; 2. Naïve B cells; 3. CD14+ Monocytes; 4. T helper cells; 5. Cytotoxic T cells; 6. Naïve Cytotoxic T cells; 7. NK cells; 8. Cytotoxic T cells; 9. Erythrocytes; 10. HSCs; 11. Pre B cells; 12. HSCs; 13. cDCs; 14. CD16+ Monocytes; 15. Pro B cells; 16. pDCs; 17. Plasma cells.

**Supplementary Figure 3. Benchmarking approximate nearest neighbor finding methods used in scCloud, SCANPY, and Seurat V3 on the bone marrow data set. a.** Boxplot on recall values for the three methods. We asked each method to report the top 100 nearest neighbors and calculated the recall value as the percentage of reported neighbors that are also in the ground truth, which is calculated using the brute force kNN algorithm. **b.** Time to complete nearest neighbor searching. We ran all methods using 28 threads and 256GB memory. scCloud ran the HNSW algorithm in full speed mode.

**Supplementary Figure 4. Diffusion map at does not efficiently use all diffusion components and thus yields less optimal developmental trajectory. a.** Bar plots visualizing diffusion coefficients of diffusion components 1 to 49 for diffusion map at and . Diffusion components with larger coefficients have higher impact on the diffusion map. When , the diffusion map is dominated by the top diffusion components. In contrast, when , all 49 diffusion components play roles in the diffusion map. **b.** Force-directed layout embedding of the developmental trajectories of the bone marrow data set using diffusion map at and . The trajectory at reveals fewer details. For example, it collapsed monocytes and conventional dendritic cells together and does not separate the two erythrocytes clusters.

**Supplementary Figure 5. Spectral community detection algorithms combine the strengths of both spectral clustering and community detection algorithms. a.** Compared to community detection algorithms (right, Leiden), spectral clustering (left) provides much faster, but less favorable clustering results. Spectral clustering tends to return clusters as evenly sized blobs. As a result, it split plasma cells and erythrocytes into multiple clusters, and merged portions of Pre-B cells, Pro-B cells, and erythrocytes into CD4+ naïve T cells. **b.** Spectral Louvain algorithm (right) is 7.5x faster than the Louvain algorithm (left) and provides clusters of comparable quality. The adjusted mutual information score (AMI) between clusters found by spectral Louvain and Louvain algorithms is 0.92.

**Code availability**

scCloud WDL source files can be found at [<https://github.com/klarman-cell-observatory/KCO>].

scCloud Terra documentation can be found at [<https://kco-cloud.readthedocs.io/en/latest/index.html>].

scCloud analysis module Python source codes can be found at [<https://github.com/broadinstitute/scRNA-Seq/tree/master/scCloud>].

scCloud command line documentation can be found at [<https://sccloud.readthedocs.io/en/latest/>].

scCloud is licensed under GPLv3.

**Data availability**

The bone marrow data set is available at [[https://preview.data.humancellatlas.org](https://preview.data.humancellatlas.org/)].

**Acknowledgments**

This publication is part of the Human Cell Atlas - [www.humancellatlas.org/publications](http://www.humancellatlas.org/publications). We thank Jennifer Rood for her help on revising this manuscript.

**Author Contributions**

B.L. and A.R. conceived the study, designed experiments and devised analyses. B.L. developed computational methods beneath scCloud. B.L. J.G., S.S. and Y.Y. implemented scCloud. B.L., J.G., S.S., Y.Y., M.T., O.A and Y.O. conducted computational experiments. M.S., M.S.K. and A.V. helped generating the immune cell atlas data used in this manuscript. T.T., E.B. and A.P. helped with the Terra cloud part of scCloud development. N.H., O.R.R. and A.R. supervised work. B.L., Y.Y. and A.R. wrote the paper with input from all the authors.

**Competing interests**

AR is a SAB member of ThermoFisher Scientific, Driver Group and Syros Pharamceuticals and a founder and consultant for Celsius Therapeutics.

**Additional information**

Supplementary Figures 1-6

**Methods**

**scCloud design.**

scCloud consists of two components: software and workflows. We encapsulate all software packages into Docker images and deposit these images into a public repository in Docker Hub [https://hub.docker.com]. We write all workflows using the Workflow Description Language (WDL, <https://github.com/openwdl/wdl>) and deposit them with public access in the Broad Methods Repository [<https://portal.firecloud.org/?return=terra#methods>].

**Gene-count matrix generation for droplet-based scRNA-Seq data.**

scCloud supports gene-count matrix generation for 10x Genomics V2 and V3 chemistry using Cell Ranger. scCloud first demultiplexes Illumina base call files (BCLs) for each sequencing flowcell by running *mkfastq* steps parallelly in different computer nodes. Each *mkfastq* job calls ‘cellranger mkfastq’ to generate sequence reads in FASTQ files. By default, each *mkfastq* job requests 64 CPUs, 128 GB memory and 1.5 TB disk space from the cloud. scCloud then generates gene-count matrices for each 10x channel by running *count* steps parallelly. Each *count* job calls ‘cellranger count’ with appropriate parameters and requests 64 CPUs, 128 GB memory and 500 GB disk space from the cloud by default.

scCloud also supports gene-count matrix generation for Drop-seq2 data using the methods described in Drop-seq alignment cookbook41.

**Gene-count matrix generation for plate-based scRNA-Seq data.**

scCloud supports gene-count matrix generation for SMART-Seq2 protocol from sequence reads in FASTQ files. scCloud estimates gene expression levels for each single cell parallelly in different computer nodes. Each node runs RSEM42 with default parameters and utilizes Bowtie 243 to align reads. Each node requests 4 CPUs, 10GB memory and 10GB disk space by default. Once expression levels are estimated, scCloud converts the relative expression levels (in Transcript per Million, TPM) into a count vector for each single cell using the formula below and then generates a gene-count matrix by concatenating count vectors from all cells.

where and are the converted read count and estimated expression level of gene , respectively. is the sum of RSEM-estimated expected counts from all genes.

**Mining of biological insights.**

scCloud runs the *analysis* step on a single computer node, which requests 64 CPUs, 200 GB memory and 100 GB disk space by default. The *analysis* step calls a super-fast Python package we implemented, which is also called scCloud. The scCloud Python package utilizes SCANPY’s AnnData data structure to store gene-count matrices and analysis results. More implementation details are discussed in the following sections and **Supplementary Note**.

**Immune Cell Atlas bone marrow dataset.**

The bone marrow dataset consists of over 270,000 cells from 8 donors. These cells were collected from 63 10x Genomics V2 chemistry chip channels. We profiled 8 channels for each donor, except for donor 6, which was profiled for only 7 channels.

**Cloud computing cost analysis.**

scCloud utilizes Google Cloud Platform’s preemptible instances. Jobs running in preemptible instances can be kicked off by others’ jobs with higher priority but are 5x cheaper. By default, scCloud allows up to 2 tries using preemptible instances before switching to non-kicked-off instances. Thus, the cost reported in the manuscript can be slightly different depending on the traffic in the cloud.

**Feature-count matrix generation for CITE-Seq, cell hashing, nucleus hashing and pooled CRISPR screens.**

scCloud supports feature-count matrix generation of CITE-Seq, cell hashing, nucleus hashing and pooled CRISPR screen protocols using either 10x Genomics V2 or V3 chemistry. Each feature-count matrix generation job runs parallelly on a separate computer node with 1 CPU, 32 GB memory and 100 GB disk space, and calls ‘generate\_count\_matrix\_ADTs’, a fast C++ program we implemented, to extract the matrix from sequence reads in FASTQ files. The C++ program scans each read pair to search for valid sequence structures. We assume read 1 records cellular barcode and Unique Molecular Identifier (UMI) information and read 2 records feature barcode information. The first 16 nucleotides of read 1 represent the cell barcode for both V2 and V3 chemistry. The next 10 and 12 nucleotides represent the UMI for V2 and V3 chemistry respectively. We allow up to 1 and 0 mismatch for matching cell barcodes in V2 and V3 chemistry respectively.

Feature barcode information is recorded differently in read 2 for different protocols. For CITE-Seq, cell hashing and nucleus hashing protocols, the location of the feature barcode depends on what type of BioLegend TotalSeqTM antibodies users choose. If TotalSeqTM-A antibodies are used, the feature barcode locates at the 5’ end of read 2 and is followed by a BAAAAAAA auxiliary sequence, where B refers to any nucleotide other than A. Otherwise, the feature barcode starts at the 11th nucleotide of 5’ end of read 2. ‘generate\_count\_matrix\_ADTs’ automatically detects antibody type by scanning read 2 of the first 1,000 read pairs and calculating the percentage of read pairs containing the auxiliary sequence. If more than 50% of read pairs contain the auxiliary sequence, we assume the antibody type is TotalSeqTM-A, otherwise it is TotalSeqTM-B or TotalSeqTM-C. We allow up to 1 mismatch for matching the auxiliary sequence.

For pooled CRISPR screen protocols, we assume that the feature barcode (protospacer) is located in front of a user-provided anchor sequence. For V2 chemistry, we first search the anchor sequence in read 2, allowing up to 2 mismatches or indels. We then extract the feature barcode at the 5’ end of the anchor sequence. For V3 chemistry, we assume users use 10x Genomics CRISPR guide capture assays and additionally check the Template Switching Oligo (TSO) sequence ‘AAGCAGTGGTATCAACGCAGAGTACATGGG’ at the 5’ end of read 2, allowing up to 3 mismatches and indels.

Once we locate the feature barcode, we match it with a user-provided white list, allowing up to 3 mismatches by default. After scanning all read pairs, ‘generate\_count\_matrix\_ADTs’ generates a feature-count matrix in CSV format: each row represents one feature, each column represents one cell barcode, and each element records the number of unique UMIs for the feature in the row in the cell barcode in the column. To speed up the sequence matching process, we encode cell barcodes, UMIs and feature barcodes into 8-byte unsigned integers (2 bits per nucleotide).

**CITE-Seq data analysis.**

We recommend that users to include both antibodies of interests, such as PD-1, and their corresponding IgG controls in their CITE-Seq assays. Based on the generated feature-count matrix, scCloud first calculates log fold change between feature UMI counts of the antibody of interest and its IgG control as the antibody expression. Let us denote the UMI counts of the antibody and its IgG control as and . The antibody expression is calculated as

We add 1 to both the numerator and denominator inside the log function to avoid the problem. Then scCloud merges the transformed antibody expression matrix into the RNA expression matrix so that users can plot antibody expressions on 2D/3D visualizations (e.g. t-SNE & UMAP) calculated based on RNA expression levels. scCloud can optionally generate t-SNE plots solely based on antibody expression levels.

**Demultiplexing cell hashing and nucleus hashing data.**

scCloud demultiplexes cell hashing and nucleus hashing data using the DemuxEM algorithm proposed in Gaublomme and Li *et al.*19

**Chimeric read filtration for pooled CRISPR screen data.**

The sgRNA library is often over-sequenced for pooled CRISPR screen assays, which results in a high number of false positive UMIs due to PCR chimeric reads44. These false positive UMIs tend to have fewer supporting reads on average. Suppose we have UMIs with exact supporting reads. In general, we expect decreases monotonically while we increase . However, if the library is over-sequenced, sometimes we can observe a second peak in the tail of the distribution (), which is more likely to represent true UMIs. scCloud detects the left boundary of the second peak by scanning consecutively. If scCloud can find an such that and , scCloud will filter out any UMIs with fewer than supporting reads. Otherwise, scCloud filters out any UMIs with only one supporting reads. If a cell barcode and UMI combination contains more than 1 feature barcode, it is likely that the feature barcode with fewer supporting reads is produced by PCR chimeras44 and scCloud will filter feature barcodes supported by no more than 10% of reads belonging to that combination. scCloud generates a filtered feature-count matrix after this filtration step and lets users decide if they want to use the original feature-count matrix or the filtered feature-count matrix.

**Immune repertoire detection and single-cell ATAC-Seq data analysis.**

scCloud supports 10x Genomics’ immune repertoire detection analysis by calling ‘cellranger vdj’ and requests 64 CPU, 128 GB memory and 500 GB disk space by default. scCloud supports 10x Genomics’ single-cell ATAC-Seq data analysis by calling ‘cellranger-atac mkfastq’ and ‘cellranger-atac count’. By default, scCloud requests 64 CPU, 128 GB memory and 1.5 TB disk space for ‘cellranger-atac mkfastq’ jobs and 64 CPU, 128 GB memory and 500 GB disk space for ‘cellranger-atac count’ jobs.

**scCloudVis implementation.**

scCloudVis is a serverless application to visualize variables on a 2D or 3D embedding of observations. The client side of scCloudVis is implemented using React to manage state and Plotly to generate charts. The backend consists of several cloud functions to manage datasets and to slice variables from a specified dataset in PARQUET format, where the PARQUET file is generated by scCloud. The slice function can optionally generate statistical summaries on an n-dimensional grid, thus enabling plotting of millions of cells.

**Curated cell-type-specific markers.**

We have curated cell-type-specific markers for immune and brain tissues in both human and mouse. These markers are collected based on domain knowledge. In particular, we collect human myeloid cell markers based on our recent paper45.

**Analysis module: preprocessing.**

scCloud first filters out low quality cells that with too little or too many detected genes or unique molecular identifiers (UMIs). It also filters out low quality cells with high percentage of UMIs from mitochondrial genes (high mitochondrial rate). scCloud then selects robust genes, which are genes detected in at least percentage of cells and normalizes the count vector of each cell such that the sum of normalized counts from robust genes is equal to 100,000 (transcript per 100K, TP100K). After that, scCloud transforms the normalized expression matrix into the natural log space by replacing expression value into . Please refer to **Supplementary Note** for more details.

**Analysis module: highly variable gene selection.**

The standard HVG selection procedure operates in the original expression space. However, almost all downstream analyses are conducted in the log expression space. To reconcile this inconsistency, we develop a new HVG selection procedure that operates directly in the log expression space.

We select HVGs only from robust genes. Suppose we have cells and robust genes. We denote the log expression of gene in cell as . We first estimate the mean and variance for each robust gene as

We then fit a LOESS46 curve of degree 2 (span parameter 0.02) between the estimated means and variances (**Supplementary Fig. 1a**) and denote the LOESS-predicted variance for gene as . Any gene with has a higher than expected variance.

We calculate the difference and fold change between the estimated and LOESS-predicted variances as

We then rank each robust gene with respect to and in descending order, and denote their rankings as and respectively. Lastly, we define the overall ranking as the sum of the two rankings

and select the top robust genes with respect to as HVGs.

The new procedure handles batch effects naturally. Suppose we have biologically different groups, each group has batches and each batch has cells. We additionally denote the mean within batch and within group as and , respectively. Now we can decompose the variance into three components: one for within-batch variance (), one for between-batch variance () and one for between-group variance ().

We remove the variance term () due to batch effects by redefining as

and plug in the new redefined variance term to the previously described procedure to select HVGs. For a detail description of the new procedure, please refer to **Supplementary Note**.

We also implements the standard HVG selection procedure, which handles batch effects using the method Seurat V3 adopts33. Please refer to **Supplementary Note** for implementation details.

**HVG selection experiments.**

We preprocessed the bone marrow data set by filtering out any cell with fewer than 500 genes or more than 6,000 genes, or with a mitochondrial rate higher than 10%. We then selected robust genes with , normalized expressions into TP100K and log-transformed the expression matrix.

We applied the standard HVG selection procedure and the new HVG selection procedure to the log-transformed expression matrix separately. We then applied the same analyses to the two sets of HVGs using scCloud with default parameters. The analyses included batch correction, dimension reduction via PCA, kNN graph construction, diffusion component calculation, community detection using the spectral-Leiden algorithm described below, 2D visualization using FIt-SNE40, trajectory visualization using the FLE algorithm, differential expression analysis and marker-based cell type annotation.

We evaluated the similarity between clusters obtained using the two HVG selection procedures using the adjusted mutual information47 (AMI) score defined below:

where and represent two cluster settings, denotes entropy and denotes mutual information.

**Analysis module: batch correction.**

scCloud corrects batch effects for all genes. For simplicity, we assume that we only have one biological group with batches and each batch has cells. We model the log gene expression level of gene at batch ’s th cell as

where is the baseline expression level of gene , is the error term, which follows a distribution of mean and variance. In addition, and are the additive and multiplicative batch effects respectively. We estimate these parameters for each gene separately as follows:

We denote as the batch adjusted expression level, which is calculated as

Since we do not expect negative log expression levels, we further set any negative adjusted expression levels to 0. Let denote the final adjusted expression level, we have

Please refer to **Supplementary Note** for a more detailed description of the L/S method and how to handle multiple biological groups.

Since batch correction transforms a sparse expression matrix into a dense matrix, which uses much more memory, we only calculate batch-adjusted expression levels for genes of interest, such as HVGs. We can rewrite (15) as

We split batch correction into two steps: First, we calculate and for all genes. Second, we calculate adjusted expressions only for genes of interest using (17).

**Benchmarking batch correction methods.**

We benchmark scCloud, ComBat, MNN, BBKNN and Seurat using a subset of the bone marrow dataset. The subset consists of the first 10x Genomics channel from each of the 8 donors. We applied the preprocessing steps described previously and obtained 34,654 cells from this subset. We then applied the new HVG selection procedure and downstream analyses described in HVG selection experiments (without batch correction) to obtain cell-type-annotated clusters (**Supplementary Fig. 2b**).

The clustering results showed strong donor-specific effects (**Supplementary Fig. 2b)**. In particular, we observed one donor-3-specific CD14+ monocyte cluster and one donor-3-specific T cell cluster. We merged the monocyte cluster into the larger monocyte cluster to its right. The donor-3-specific T cell cluster is adjacent to 5 T cell clusters. Thus, we trained a LightGBM28 classifier that predicts cluster label using expression levels using cells from the 5 clusters (90% training data + 10% validation data). The classifier has a validation accuracy of 88.4%. We then used this classifier to assign each cell in the donor-3-specific T cell cluster into one of the 5 adjacent T cell clusters.

The annotated cell types are used as ground truth to evaluate different batch correction methods. A good batch correction method should 1) mix cells from 8 donors well and 2) keep cells with same annotated types close to each other. We use kBET and kSIM acceptance rates to assess these two criteria.

To generate batch-corrected results for benchmarking, we applied the same preprocessing step to the subset and selected top 2000 HVGs using the new HVG selection procedure (with batch effects handled). We obtained scCloud-corrected expression levels using the L/S adjustment method. We then extracted the HVG-specific gene-count matrix and fed it to SCANPY to obtain ComBat-corrected and MNN-corrected expression levels. We fed the matrix to Seurat V3 to obtain Seurat-corrected expression levels. We performed PCA, kNN graph construction, and UMAP on using the corrected expression matrices. For BBKNN correction, we used BBKNN to replace scCloud’s kNN graph construction and kept other analyses the same.

**kBET acceptance rate.**

kBET34 acceptance rate measures if cells from different batches mix well in the local neighborhood of each cell. scCloud implements kBET acceptance rate calculation in its analysis module. We define as the ideal batch mixing frequency, where . For each cell , we find its nearest neighbors (including itself) using the HNSW algorithm35 and denote the number of neighbors belong to batch as . Then we calculate its test statistic with degrees of freedom as

and its p value as

where is the cumulative density function.

kBET acceptance rate is the percentage of cells that accept the null hypothesis at significant level :

where is the indicator function.

To benchmark different batch correction methods, we calculate kBET acceptance rates based on UMAP 2D coordinates and set .

**kSIM acceptance rate.**

kSIM acceptance rate is inspired by kBET. It requires ground truth cell type information and measures if the neighbors of a cell have the same cell type as this cell. If a method over-corrects the batch effects, it will have a low kSIM acceptance rate. We use the HNSW algorithm to find nearest neighbors (including the cell itself) for each cell and denote the number of neighbors that have the same cell type as as . In addition, we require at least fraction of neighbors of cell to have the same cell type as in order to say cell has a consistent neighborhood. The kSIM acceptance rate is calculated as follows:

We calculate kSIM acceptance rate based on UMAP 2D coordinates as well and set .

**Analysis module: dimension reduction (Principal Component Analysis).**

scCloud calculates the top principal components based on highly variable genes. It utilizes the randomized PCA algorithm48 implemented in *Scikit-learn* package49 to speed up the computation. By default, scCloud sets .

**Analysis module: K nearest neighbor graph construction.**

scCloud uses the HNSW35 algorithm with parameters , , , to construct kNN graphs. By default, scCloud searches the top nearest neighbors (including the cell itself) for each cell (). Because HNSW is an approximate algorithm, it cannot always return the cell itself as the 1st nearest neighbor. For any cell missing itself as the 1st nearest neighbor, scCloud sets itself as the 1st nearest neighbor and picks the top 99 nearest neighbors returned by HNSW as the 2nd to 100th nearest neighbors. HNSW has a random index building process, which produces different indices in different runs if multiple threads are used. For reproducibility purpose, scCloud provides two modes of running HNSW: robust mode and full speed mode. In robust mode, scCloud runs the index building process with only 1 thread and runs the neighbor searching process with multiple threads. In full speed mode, scCloud also runs the index building process with multiple threads. In either mode, scCloud stores the neighbor searching results in the AnnData object. Without explicit notification, scCloud runs HNSW in the robust mode.

**Benchmarking approximate nearest neighbor finding methods.**

We benchmarked the approximate nearest neighbor finding algorithms used by scCloud, SCANPY and Seurat on the bone marrow dataset with default parameters. scCloud runs the HNSW algorithm in full speed mode, SCANPY uses the algorithm implemented in UMAP, and Seurat uses the RANN package50. We ran the three methods on coordinates from top 50 PCs produced by scCloud and sought for top 100 nearest neighbors (including the cell itself). We also ran the brute force kNN searching algorithm using scikit-learn49 to compute the ground truth. We ran all methods on a computer server with 28 threads and 256G memory. We evaluate each method’s performance using recall, defined as the percentage of K nearest neighbors that are also in the ground truth, and speed.

**Analysis module: diffusion map and diffusion pseudotime**

scCloud improves upon the DPT37 approach to calculate diffusion map and diffusion pseudotime. We only provide a high-level summarization of our methods here and please refer to **Supplementary Note** for detailed mathematical derivations.

scCloud first constructs an affinity matrix based on top principal components. This affinity matrix is also used in community-detection-based clustering algorithms. To construct , we need to find top nearest neighbors for each cell. Our first improvement over DPT is the use of the HNSW algorithm to find top nearest neighbors. With the nearest neighbors, we can define the following locally-scaled Gaussian kernel between any two cells and :

In the above equation, is the vector containing the top PC coordinates for the cell and is its local kernel width defined as , where is the distance between this cell and its th neighbor. To eliminate the effects of sampling density, we additionally define the following density-normalized kernel21:

is the sampling density term for cell . Let us define as the set consisting of ’s 2nd to th neighbors, is calculated as follows:

We construct using the density-normalized kernel as follows:

scCloud then calculates the Markov chain transition matrix and symmetric “transition” matrix based on the affinity matrix:

Since is symmetric, it has the eigen decomposition of . In addition, we know that all ’s eigenvalues are in and is its eigenvector for eigenvalue (**Supplementary Note**). We also know that share the same eigenvectors as and its right eigenvectors are

Oursecond improvement is to calculate the top eigenvalues of using the randomized SVD algorithm48 (**Supplementary Note**). We order eigenvalues by magnitude . By default, scCloud sets .

We now define a family of traditional diffusion maps parameterized by time :

Note that unlike scCloud, DPT uses , eigenvectors of in its diffusion maps.

Instead of picking a value for , we want an embedding that can combine diffusion maps at a variety of timescales. Therefore, we define a new family of diffusion maps :

and define as the diffusion coefficient for diffusion component .

This new family of diffusion maps is our third improvement. In particular, when , we recover the embedding that DPT uses to calculate diffusion pseudotime. scCloud uses by default.

Once user picks a cell as the root, we can calculate the diffusion distance from root to any other cell as

We then normalize the diffusion distance into and use the normalized distance as the diffusion pseudotime.

**Comparison between diffusion maps at and**

We ran diffusion maps at and on the bone marrow data set. For , we processed data exactly the same as in the HVG selection experiments. For , we construct diffusion map with using the PC components and kNN graph calculated for and then generated force-directed layout embedding using the resulting diffusion map. We colored the embedding using the same cell type annotation generated for .

**Analysis module: modularity-based community detection algorithms.**

Modularity-based community detection algorithms work on graphs. In scCloud, we construct a weighted undirected graph from the affinity matrix . In the graph, vertex set contains all cells, and an edge if and only if . The weight of the edge is calculated as follows:

These algorithms try to find a partition of cells that maximizes the modularity function51 below:

where each contains cells in that community, is the resolution parameter controlling the total number of communities, and

scCloud supports two modularity-based community detection algorithms: Louvain23 and Leiden24. For both algorithms, scCloud sets the resolution by default. scCloud reports each community as a separate cluster.

The Louvain algorithm optimizes the modularity function in two phases: (1) move phase, each node is inspected and moved to the community that yield the largest increase in ; (2) aggregation phase, each community aggregates into a new node to form an aggregated graph. This algorithm starts from the partition that each cell is its own community and repeats the two phases until there is no increase in . scCloud uses the louvain-igraph [<https://github.com/vtraag/louvain-igraph>] implementation from Vincent Traag. Note that the latest release of louvain-igraph package (v0.6.1) contains a bug that prevent it from being reproducible even when the same random seed is used. Thus, scCloud installs this package directly from the master git branch, which has the bug fixed.

The Leiden algorithm is a recent improvement over the Louvain algorithm and consists of three phases: (1) move phase, which is similar to Louvain’s; (2) refinement phase, each community found in (1) is examined and may be split into multiple sub-communities; (3) aggregation phase, each sub-community from (2) is aggregated into a new node and assigned to an initial partition based on communities from (1). scCloud uses the leidenalg [<https://github.com/vtraag/leidenalg>] implementation from Vincent Traag. Applying the Leiden algorithm on communities detected from previous Leiden runs can further improve the modularity function24. Thus, following SCANPY, scCloud runs the Leiden algorithm iteratively on the graph until is not further improved (n\_iterations = -1).

**Analysis module: spectral-community-detection algorithms for fast clustering.**

scCloud provides two spectral-community-detection algorithms: spectral-Louvain and spectral-Leiden. These algorithms combine the strengths of both spectral clustering and community detection algorithms --- they scale well to large data sets and produce clustering results comparable to their corresponding community detection algorithm. Spectral community detection algorithms run in two steps: (1) spectral clustering is used to partition cells into tens of thousands robust groups; (2) each group of cells is aggregated into one node and then either Louvain or Leiden algorithm is applied to the aggregated graph, which is much smaller than the original graph. We describe how step (1) works below.

Applying the k-means algorithm on diffusion components gives us the spectral clustering39 algorithm. It is well known that the k-means clustering results highly depend on the initial cluster centers. If we run k-means multiple times with different random initial cluster centers, we will get different clustering results. Suppose we run k-means times, each time finds clusters and we denote the clustering results of th run as , which is a function mapping each cell to its cluster. We can partition cells into robust groups such that any two cells in the same robust group must be in a same cluster for all k-means runs:

Since cells in the same robust group are always clustered together in all runs, they are highly likely to come from a same true cluster. scCloud uses and by default.

**Analysis module: t-SNE, UMAP and force-directed layout embedding.**

scCloud uses the Multicore-TSNE package [<https://github.com/DmitryUlyanov/Multicore-TSNE>] implemented by Dmitry Ulyanov to calculate t-SNE embedding. We found and fixed a random-seed-related bug in this package that prevents the package to reproduce the exact t-SNE coordinates. The bug fixed package is available at [<https://github.com/bli25broad/Multicore-TSNE>]. scCloud uses the pyFIt-SNE package [<https://github.com/KlugerLab/pyFIt-SNE>] to calculate FIt-SNE embedding, which is a fast approximation of the t-SNE embedding. scCloud uses the umap package [https://github.com/lmcinnes/umap] to calculate UMAP embedding.

**Analysis module: deep-learning-based visualization.**

**References**

1. Regev, A. *et al.* The Human Cell Atlas White Paper. *arXiv:1810.05192 [q-bio]* (2018).

2. Macosko, E. Z. *et al.* Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell* **161**, 1202–1214 (2015).

3. Rosenberg, A. B. *et al.* Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science* **360**, 176–182 (2018).

4. 10x Genomics. *Transcriptional Profiling of 1.3 Million Brain Cells with the Chromium Single Cell 3’ Solution*. (2017).

5. Cao, J. *et al.* The single-cell transcriptional landscape of mammalian organogenesis. *Nature* **566**, 496–502 (2019).

6. Birger, C. *et al.* FireCloud, a scalable cloud-based platform for collaborative genome analysis: Strategies for reducing and controlling costs. *bioRxiv* 209494 (2017). doi:10.1101/209494

7. Zheng, G. X. Y. *et al.* Massively parallel digital transcriptional profiling of single cells. *Nat Commun* **8**, 14049 (2017).

8. Satija, R., Farrell, J. A., Gennert, D., Schier, A. F. & Regev, A. Spatial reconstruction of single-cell gene expression data. *Nat. Biotechnol.* **33**, 495–502 (2015).

9. Human Immune Cell Atlas Project Data. Available at: https://preview.data.humancellatlas.org.

10. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* **19**, 15 (2018).

11. Picelli, S. *et al.* Smart-seq2 for sensitive full-length transcriptome profiling in single cells. *Nat. Methods* **10**, 1096–1098 (2013).

12. Stoeckius, M. *et al.* Simultaneous epitope and transcriptome measurement in single cells. *Nat. Methods* **14**, 865–868 (2017).

13. Dixit, A. *et al.* Perturb-Seq: Dissecting Molecular Circuits with Scalable Single-Cell RNA Profiling of Pooled Genetic Screens. *Cell* **167**, 1853-1866.e17 (2016).

14. Adamson, B. *et al.* A Multiplexed Single-Cell CRISPR Screening Platform Enables Systematic Dissection of the Unfolded Protein Response. *Cell* **167**, 1867-1882.e21 (2016).

15. Jaitin, D. A. *et al.* Dissecting Immune Circuits by Linking CRISPR-Pooled Screens with Single-Cell RNA-Seq. *Cell* **167**, 1883-1896.e15 (2016).

16. Datlinger, P. *et al.* Pooled CRISPR screening with single-cell transcriptome readout. *Nat. Methods* **14**, 297–301 (2017).

17. Gasperini, M. *et al.* A Genome-wide Framework for Mapping Gene Regulation via Cellular Genetic Screens. *Cell* **176**, 377-390.e19 (2019).

18. Stoeckius, M. *et al.* Cell Hashing with barcoded antibodies enables multiplexing and doublet detection for single cell genomics. *Genome Biol.* **19**, 224 (2018).

19. Gaublomme, J. T. *et al.* Nuclei multiplexing with barcoded antibodies for single-nucleus genomics. *bioRxiv* 476036 (2018). doi:10.1101/476036

20. Tabaka, M., Gould, J. & Regev, A. scSVA: an interactive tool for big data visualization and exploration in single-cell omics. *bioRxiv* 512582 (2019). doi:10.1101/512582

21. Coifman, R. R. & Lafon, S. Diffusion maps. *Applied and Computational Harmonic Analysis* **21**, 5–30 (2006).

22. Haghverdi, L., Büttner, M., Wolf, F. A., Buettner, F. & Theis, F. J. Diffusion pseudotime robustly reconstructs lineage branching. *Nature Methods* **13**, 845 (2016).

23. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech.* **2008**, P10008 (2008).

24. Traag, V. A., Waltman, L. & Eck, N. J. van. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* **9**, 5233 (2019).

25. Maaten, L. van der & Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research* **9**, 2579–2605 (2008).

26. McInnes, L., Healy, J. & Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. (2018).

27. 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* **9**, e98679 (2014).

28. Ke, G. *et al.* LightGBM: A Highly Efficient Gradient Boosting Decision Tree. in *Advances in Neural Information Processing Systems 30* 3146–3154 (2017).

29. Li, C. & Wong, W. H. DNA-Chip Analyzer (dChip). in *The Analysis of Gene Expression Data* 120–141 (2003).

30. Johnson, W. E., Li, C. & Rabinovic, A. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* **8**, 118–127 (2007).

31. Haghverdi, L., Lun, A. T. L., Morgan, M. D. & Marioni, J. C. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nature Biotechnology* **36**, 421 (2018).

32. Park, J.-E., Polański, K., Meyer, K. & Teichmann, S. A. Fast Batch Alignment of Single Cell Transcriptomes Unifies Multiple Mouse Cell Atlases into an Integrated Landscape. *bioRxiv* 397042 (2018). doi:10.1101/397042

33. Stuart, T. *et al.* Comprehensive Integration of Single-Cell Data. *Cell* **177**, 1888-1902.e21 (2019).

34. Büttner, M., Miao, Z., Wolf, F. A., Teichmann, S. A. & Theis, F. J. A test metric for assessing single-cell RNA-seq batch correction. *Nature Methods* **16**, 43 (2019).

35. Malkov, Y. A. & Yashunin, D. A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1–1 (2018). doi:10.1109/TPAMI.2018.2889473

36. Aumüller, M., Bernhardsson, E. & Faithfull, A. ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms. in *Similarity Search and Applications* 34–49 (Springer, Cham, 2017). doi:10.1007/978-3-319-68474-1\_3

37. Haghverdi, L., Büttner, M., Wolf, F. A., Buettner, F. & Theis, F. J. Diffusion pseudotime robustly reconstructs lineage branching. *Nature Methods* **13**, 845 (2016).

38. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech.* **2008**, P10008 (2008).

39. Ng, A. Y., Jordan, M. I. & Weiss, Y. On spectral clustering: analysis and an algorithm. in 849–856 (MIT Press, 2001).

40. Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S. & Kluger, Y. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods* **16**, 243 (2019).

41. Nemesh, J. Drop-seq Alignment Cook Book. Available at: https://github.com/broadinstitute/Drop-seq/blob/master/doc/Drop-seq\_Alignment\_Cookbook.pdf.

42. Li, B. & Dewey, C. N. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics* **12**, 323 (2011).

43. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357–359 (2012).

44. Dixit, A. Correcting Chimeric Crosstalk in Single Cell RNA-seq Experiments. *bioRxiv* 093237 (2016). doi:10.1101/093237

45. Villani, A.-C. *et al.* Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science* **356**, eaah4573 (2017).

46. Cleveland, W. S., Grosse, E. & Shyu, W. M. Local Regression Models. in *Statistical Models in S* (1992). doi:10.1201/9780203738535-8

47. Vinh, N. X., Epps, J. & Bailey, J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? in 1073–1080 (Proceedings of the 26th Annual International Conference on Machine Learning, 2009). doi:10.1145/1553374.1553511

48. Halko, N., Martinsson, P. G. & Tropp, J. A. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review* (2011). doi:10.1137/090771806

49. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).

50. Arya, S., Kemp, S. E., Jefferis, G. & Mount, D. *RANN: Fast Nearest Neighbour Search (Wraps ANN Library) Using L2 Metric*. (2019).

51. Reichardt, J. & Bornholdt, S. Statistical mechanics of community detection. *Phys. Rev. E* **74**, 016110 (2006).