nature methods | BRIEF COMMUNICATION
https://doi.org/10.1038/s41592-019-0353-7

Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning

Yue Deng[1,3], Feng Bao[2,3], Qionghai Dai[2], Lani F. Wu[1]* and Steven J. Altschuler[1]*

Recent advances in large-scale single-cell RNA-seq enable fine-grained characterization of phenotypically distinct cellular states in heterogeneous tissues. We present scScope, a scalable deep-learning-based approach that can accurately and rapidly identify cell-type composition from millions of noisy single-cell gene-expression profiles.

Single-cell RNA-seq (scRNA-seq) provides high-resolution dissection of complex biological systems, including identification of rare cell subpopulations in heterogeneous tissues, elucidation of cell-developmental programs, and characterization of cellular responses to perturbations[1–3]. Recent scRNA-seq experimental platforms[4–6] have enabled interrogation of millions of cells, offering an unprecedented resolution at which to dissect cell-type compositions.

These advances have led to two acute challenges. First, single-cell transcriptomics is susceptible to amplification noise and dropout events[7,8], which become more pronounced as tradeoffs are made to sequence larger numbers of cells. Second, computational memory and/or speed restrictions may render analytical packages poorly scalable for large datasets[7–12].

To extract informative representations from extremely noisy, massive, high-dimensional single-cell RNA profiles, we developed scScope, a deep-learning-based software package (Fig. 1a). scScope uses a recurrent network layer to iteratively perform imputations on zero-valued entries of input scRNA-seq data (Methods). scScope's architecture allows imputed output to be iteratively improved through a selected number of recurrent steps (T). We note that for $T=1$, the architecture reduces to a standard autoencoder[13]. In one joint framework, scScope conducts batch effect removal, cellular-feature learning, dropout imputation and parallelized training (when multiple graphics processing units (GPUs) are available) (Supplementary Fig. 1).

We evaluated scScope on its scalability, ability to identify cell subpopulations, impute dropout gene information and correct batch effects using datasets with varying degrees of size, complexity and previous biological annotation. We compared scScope performance with seven approaches (principal component analysis (PCA), MAGIC[11], ZINB-WaVE[8], SIMLR[9] and three 'deep' learning methods: autoencoder (AE), single-cell variational inference (scVI)[14] and the deep count autoencoder (DCA)[15]) (Methods and Supplementary Table 1). The autoencoder was implemented via scScope with $T=1$ (Methods) and, for all comparisons, unless otherwise stated, scVI, DCA, autoencoder and scScope were run on a single GPU using default parameters (Methods and Supplementary Table 2).

We first tested the scalability and training speed of scScope on a mouse brain dataset that contained 1.3 million cells, using 1,000 highly variable genes (Fig. 1b and Methods). As expected, the generic machine-learning tool PCA was the fastest. scScope was able to complete its analysis of the full dataset in under an hour, which was comparable to the autoencoder, and this runtime was greatly decreased by use of the option for multiple GPU training. The single-cell software packages MAGIC, ZINB-WaVE and SIMLR were unable to scale beyond 100,000 cells; the deep approaches, while able to scale to 1 million cells, required at least seven times more computing time than scScope. Thus, scScope offers a scalable and highly efficient approach for analyzing large scRNA-seq datasets.

To calibrate the accuracy of scScope on simulated datasets, we made use of two third-party packages for generating scRNA-seq data (Methods and Supplementary Table 3). First, we used Splatter[16] to generate moderate-sized datasets of varying sparsity levels (percentage of 0-valued genes), containing 2,000 scRNA-seq profiles with 500 genes and three underlying subpopulations. In terms of discovering these underlying subpopulations, scScope performed similarly to other approaches when there were only minor dropout effects, but showed a large advantage in accuracy as dropout rates increased to ranges observed in biological data[4,6] (Fig. 1c). In terms of imputation error, at low sparsity (<50%) scScope and MAGIC outperformed all other methods, whereas at high sparsity scScope performed best (Supplementary Fig. 2). We found for scScope that a recurrence of $T=2$ (its default value) provided a good tradeoff between speed and accuracy (Supplementary Fig. 3).

Second, we used the simulation framework in SIMLR to generate massive, heterogeneous datasets of varying sparsity levels containing 1 million scRNA-seq profiles with 500 genes and 50 underlying subpopulations. The deep approaches were able to operate over the full datasets, whereas the other single-cell packages required downsampled training strategies (Methods). scScope performed well, particularly at high sparsity levels (Supplementary Fig. 4). An increasingly important task for scRNA-seq profiling approaches is to identify rare cell subpopulations in large-scale data. As might be expected, methods that did not require downsampling were better able to detect rare cell subpopulations. Overall, scScope performed comparatively well on this challenging task (Fig. 1d and Methods). Our analyses provide callibration for scScope's ability to efficiently identify cell subpopulations from complex datasets with high dropout rates, large numbers of subpopulations and rare cell types.

We next evaluated scScope on four experimental single-cell RNA datasets containing varying degrees of biological 'ground truth'.

[1]Department of Pharmaceutical Chemistry, University of California, San Francisco, San Francisco, CA, USA. [2]Department of Automation, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China. [3]These authors contributed equally: Yue Deng, Feng Bao. *e-mail: lani.wu@ucsf.edu; steven.altschuler@ucsf.edu

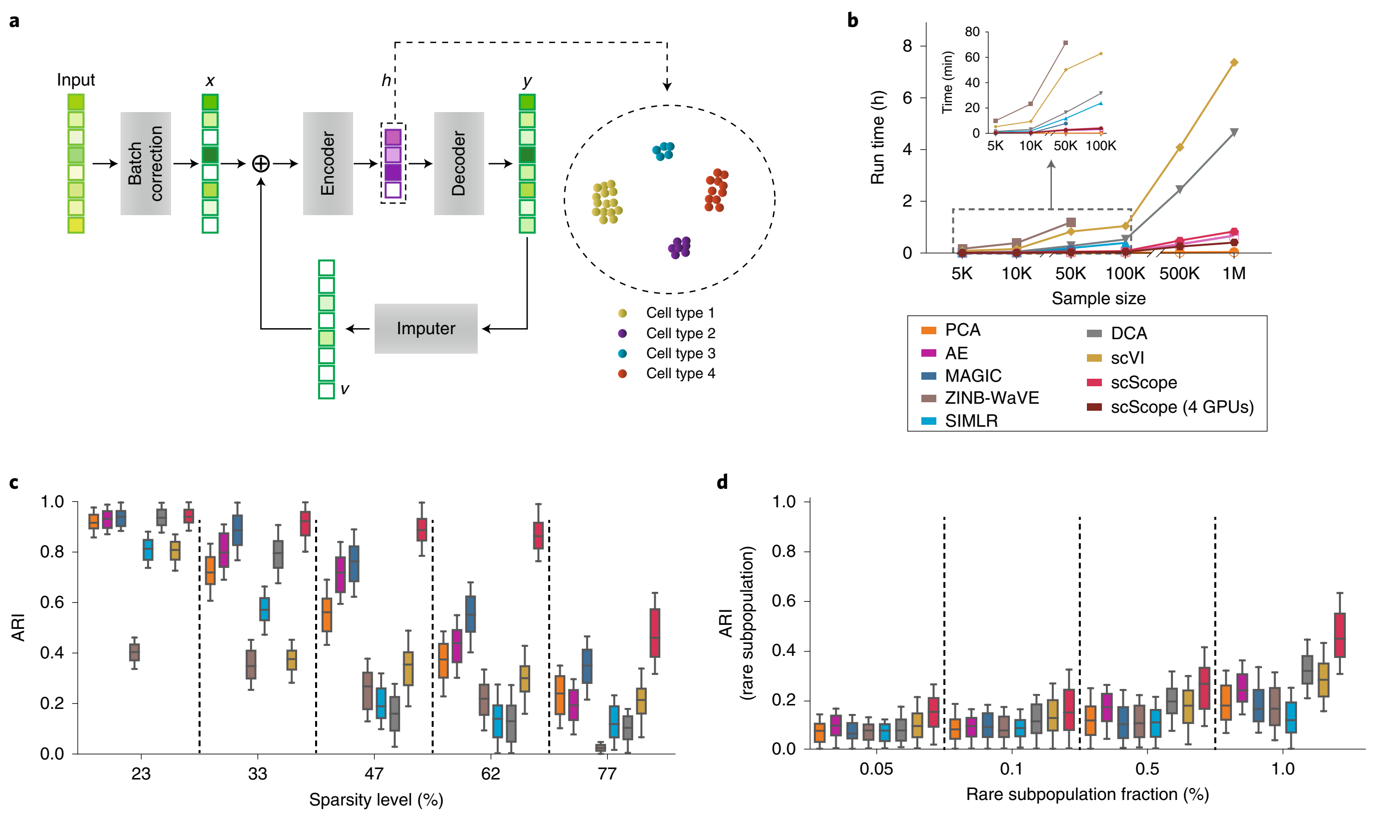

Fig. 1 | Overview of scScope architecture and performance on simulated datasets. a, Recurrent network architecture of scScope. Schematic overview. **b**, Comparison of runtimes on datasets of different scales. Datasets of varying size were randomly subsampled from a dataset containing 1.3 million mouse brain cells and used for comparison (Methods). M, million; K, thousand. **c**,**d**, Clustering accuracy for 2,000 scRNA-seq data with varying fractions of sparsity (**c**) and clustering accuracy for 1 million scRNA-seq data with varying fractions of rare subpopulations (**d**). **c**, A total of 2,000 cells were simulated with Splatter; accuracy measurement is based on an adjusted Rand index (Methods). For each simulated condition, $n=10$ random replicates were simulated. Box plot: median (center line), interquartile range (box) and minimum–maximum range (whiskers). **d**, A total of 1 million cells were simulated with single-cell interpretation via multi-kernel learning (SIMLR). Dropout rate, 0.5; total number of clusters, 50; number of rare subpopulations, 5; replicate number $n=10$. For Markov affinity-based graph imputation of cells (MAGIC), zero-inflated negative binomial-based wanted variation extraction (ZINB-WaVE) and SIMLR, the 1 million datasets were randomly downsampled to 20,000, and PhenoGraph[20] was used for de novo cell subpopulation discovery. For methods run on the 1 million dataset, a scalable clustering approach was used to identify subpopulations (Methods). Box plot as in **c**.

These datasets were used to test the ability of scScope to remove batch effects (Fig. 2a; lung tissue), recover dropout genes (Fig. 2b; cord blood mononuclear cell (CBMC) dataset), identify minor subpopulations (Supplementary Fig. 5; retina dataset) and test clustering accuracy for a large dataset with varying numbers of analyzed genes (Fig. 2c; mouse cell atlas).

To test the ability to remove batch effects, we made use of the lung tissue dataset (part of the mouse cell atlas), which contained ~7,000 scRNA-seq profiles obtained from three different batches and used the 1,000 most variable genes (Methods). As with the two other software packages that offer batch correction, scScope performed well (Fig. 2a, top) without compromising its fast runtime (Fig. 2a, bottom).

To investigate how imputation accuracy depends on gene-expression level, we made use of the CBMC dataset, with ~8,000 scRNA-seq profiles and 1,000 most variable genes (Methods). We sequentially simulated dropouts for genes[14] (Methods) and reported results based on octile of expression ranking (Fig. 2b). For reconstructing small count values, MAGIC and scVI performed well, while for large count values DCA worked well. scScope showed small imputation errors consistently across the entire range of expression.

To test the ability to identify minor subpopulations in biological data, we made use of the mouse retina dataset, with 44,000 cells and 384 most variable genes (Methods). scScope automatically identified the most similar clustering (number and assignment) to the 39 cell subpopulations identified in the original study (Supplementary Fig. 5a). We annotated the clusters to cell types on the basis of gene markers reported in the original study (Supplementary Table 4 and Methods). Clusters identified by scScope showed the most statistically significant enrichment of specific cell-type markers (larger fold changes) (Supplementary Fig. 5b) and were highly consistent with previous, microscopy-validated estimates of cell-type composition and proportion[17] either excluding (Supplementary Fig. 5c) or including (Supplementary Fig. 5d) the main cell type, rod cells. Additionally, when analyzing 'pure' cell types in the retina dataset, scScope maintained the underlying simple subpopulation structure and achieved high reconstruction accuracy (Supplementary Table 5).

To test the ability to analyze increasing numbers of genes, we made use of the mouse cell atlas, with 400,000 cells sampled from 51 tissues. Only the deep-learning algorithms were able to scale to these data sizes. To perform automatic identification of subpopulations on large datasets, we designed a scalable clustering approach (Methods and Supplementary Fig. 6). We used the 51 known tissue types to assess accuracy of the clustering results. For 1,000 and 2,000 genes, all three approaches performed similarly (Fig. 2c). Only scScope was able to scale to levels above 10,000 genes, by using its option for scalable memory allocation (Methods).

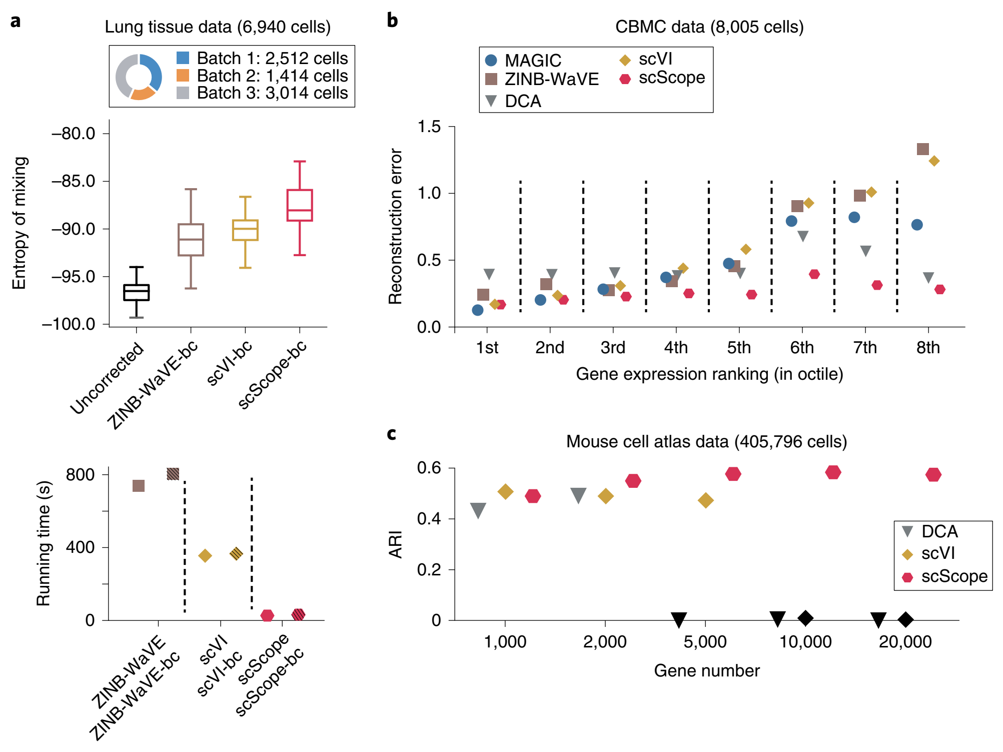

Fig. 2 | Evaluation of methods on experimental scRNA-seq datasets. a, Analysis of batch correction. Comparisons of (top) batch mixing entropy and (bottom) computational runtime without or with batch correction using mouse lung tissue scRNA-seq dataset. Top box plot: median (center line), interquartile range (box) and minimum–maximum range (whiskers); $n = 100$ replicates of 100 randomly selected cells across all batches. Bottom: runtime to process the whole dataset. **b**, Analysis of imputation accuracy for different gene-expression levels. Comparison of imputation error for dropout genes with different (octiles) gene-expression levels using the CBMC scRNA-seq dataset. **c**, Analysis of subpopulation identification for increasing gene depth. We compared the ability of different approaches to identify the 51 known tissues in the atlas. Black color: the provided software package was unable to complete the task.

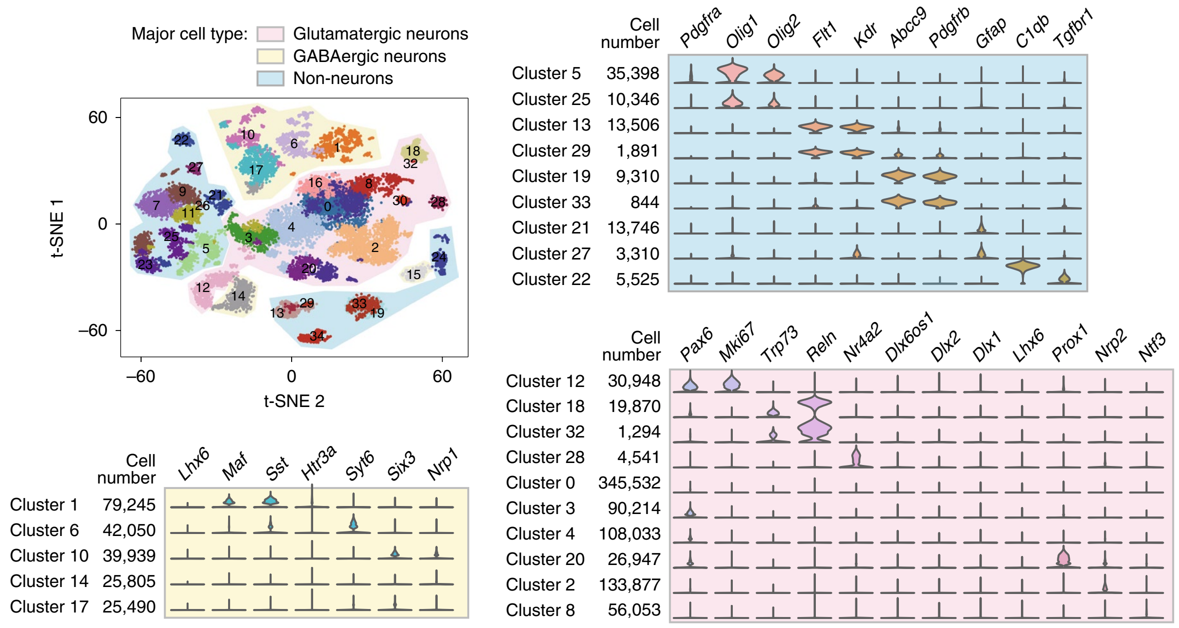

Fig. 3 | Application of scScope to explore biology in 1.3 million mouse brain dataset. scScope results visualized using t-SNE on $n = 30,000$ cells (randomly sampled from the full dataset). Clusters were divided into three main types on the basis of gene markers. Assignment of clusters to known cell types is based on top ten overexpressed genes (Supplementary Table 9). Violin plots: expression distribution of marker genes for discovered clusters. Vertical axis: clusters with known cell-type annotations and corresponding cell numbers. Horizontal axis: differentially expressed marker genes across shown clusters.

Finally, we applied scScope to investigate novel biology in datasets. We focused on the ability to reveal changes in cell-type composition under perturbed conditions (Supplementary Fig. 7; intestinal dataset) and scale to large datasets and reveal new subpopulations (Fig. 3; brain dataset and Methods).

We applied scScope to the intestinal dataset, with ~10,000 mouse intestinal cells obtained from different infection conditions and the 1,000 most variable genes (Supplementary Fig. 7a and Methods). In the original study, enterocytes were identified as a single cluster. scScope subdivided this cell type into four subpopulations: differential expression of markers delineated distal versus proximal enterocyte subpopulations; expression levels of these markers delineated immature versus mature subtypes (Supplementary Figs. 7b and 8). These refined enterocyte subpopulations suggested predictions about specific cell-type response to infection. For example, the pro-inflammatory gene *Saa1* was overexpressed during both *Salmonella* and *Heligmosomoides polygyrus* (day 10) infections in distal enterocytes, but not in proximal enterocytes (Supplementary Fig. 7c and Supplementary Table 6). This geographic pattern of *Saa1* expression is known for *Salmonella* infection, but is a novel prediction for *H. polygyrus* infection. Thus, scScope can be used to rapidly explore scRNA-seq data across perturbed conditions to predict novel gene function and identify new cell subtypes.

The 1.3 million cells in the brain dataset were obtained from multiple brain regions, including the cortex, hippocampus and ventricular zones, of two embryonic mice. Here, we made use of scScope's ability to rapidly explore this large dataset via its multi-GPU learning functionality with limited training iterations (Methods and Supplementary Table 2). scScope automatically identified 36 clusters, and we assigned each cluster to one of three main cell types on the basis of criteria from the Allen Brain Atlas (http://brain-map.org) (Fig. 3, Supplementary Table 7 and Methods): glutamatergic neurons, GABAergic neurons and non-neuronal cells. The proportions of neurons and non-neurons identified by scScope were consistent with cell proportions reported by a previous brain study[18] (Supplementary Fig. 9). We investigated whether we could identify biological meaning for the 36 clusters, some of which contained fewer than 1,000 cells. Satisfyingly, by comparing our top overexpressed genes with known cell-type markers[18,19] (Fig. 3 and Supplementary Table 8), we were able to assign two-thirds of the clusters to known cell types (Supplementary Table 9). Thus, scScope is able to rapidly, automatically and directly identify bona fide, rare cell types from large and complex biological datasets.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, statements of data availability and associated accession codes are available at https://doi.org/10.1038/s41592-019-0353-7.

Received: 11 October 2018; Accepted: 12 February 2019;
Published online: 18 March 2019

References

1. Gawad, C., Koh, W. & Quake, S. R. *Nat. Rev. Genet.* **17**, 175–188 (2016).
2. Saliba, A.-E., Westermann, A. J., Gorski, S. A. & Vogel, J. *Nucleic Acids Res.* **42**, 8845–8860 (2014).
3. Shalek, A. K. et al. *Nature* **510**, 363–369 (2014).
4. Macosko, E. Z. et al. *Cell* **161**, 1202–1214 (2015).
5. Zheng, G. X. Y. et al. *Nat. Commun.* **8**, 14049 (2017).
6. Han, X. et al. *Cell* **172**, 1091–1107 (2018).
7. Pierson, E. & Yau, C. *Genome. Biol.* **16**, 241 (2015).
8. Risso, D., Perraudeau, F., Gribkova, S., Dudoit, S. & Vert, J.-P. *Nat. Commun.* **9**, 284 (2018).
9. Wang, B., Zhu, J., Pierson, E., Ramazzotti, D. & Batzoglou, S. *Nat. Methods* **14**, 414–416 (2017).
10. Cleary, B., Le, C., Cheung, A., Lander, E. S. & Regev, A. *Cell* **171**, 1424–1436 (2017).
11. van Dijk, D. et al. *Cell* **174**, 716–729 (2018).
12. Butler, A., Hoffman, P., Smibert, P., Papalexi, E. & Satija, R. *Nat. Biotechnol.* **36**, 411–420 (2018).
13. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. *J. Mach. Learn. Res.* **11**, 3371–3408 (2010).
14. Lopez, R., Regier, J., Cole, M. B., Jordan, M. I. & Yosef, N. *Nat. Methods* **15**, 1053–1058 (2018).
15. Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S. & Theis, F. J. *Nat. Commun.* **10**, 390 (2019).
16. Zappia, L., Phipson, B. & Oshlack, A. *Genome. Biol.* **18**, 174 (2017).
17. Jeon, C. J., Strettoi, E. & Masland, R. H. *J. Neurosci.* **18**, 8936–8946 (1998).
18. Rosenberg, A. B. et al. *Science* **360**, 176–182 (2018).
19. Tasic, B. et al. *Nat. Neurosci.* **19**, 335–346 (2016).
20. Levine, J. H. et al. *Cell* **162**, 184–197 (2015).

Acknowledgements

We thank J. Chang, S. Rajaram, L. Sanman and S. Shen for their helpful comments. We gratefully acknowledge the support of NIH grant numbers R01 EY028205 and GM112690, NSF PHY-1545915 and SU2C/MSKCC 2015-003 to S.J.A., NCI-NIH no. RO1 CA185404 and CA184984 to L.F.W., the Institute of Computational Health Sciences (ICHS) at UCSF to S.J.A. and L.F.W., and Project of NSFC (no. 61327902), Project of Beijing Municipal Science & Technology Commission (no. Z181100003118014) to Q.D. and F.B.

Author contributions

Y.D., F.B. and Q.D. developed the deep-learning algorithms. Y.D. and F.B. conducted experimental analysis on both simulated and biological datasets. The manuscript was written by Y.D., F.B., L.F.W. and S.J.A. All authors read and approved the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41592-019-0353-7.

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence and requests for materials should be addressed to L.F.W. or S.J.A.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Methods

scScope model and training. *Architecture.* The architecture of the scScope network has four modules (Fig. 1a). The parameters in these layers are learned from data in an end-to-end manner through optimization. We note that scScope is flexible in terms of normalizing and scaling of input data, as long as the input values are non-negative.

Batch effect correction. scScope offers the option to correct for batch effects, inspired by a previously developed approach[8]. The batch correction layer $f_B(\cdot)$ is given by

$$x_c = f_B(\tilde{x}_c) = r(\tilde{x}_c - Bu_c)$$

Here, we denote the input single-cell profile by $\tilde{x}_c \in \mathbb{R}^N$; the number of batches by K; the binary experimental batch effects indicator vector $u_c \in \mathbb{R}^K$ (non-zero entry indicates the batch of $\tilde{x}_c$); and the learnable batch correction matrix as $B \in \mathbb{R}^{N\times K}$. Throughout, $r(\cdot)$ denotes the standard rectified linear unit (ReLU), $r(v) = \max(0, v)$; the ReLU enforces $x_c \geq 0$, which is expected for actual gene count data, and is widely used in deep learning. By default, $u_c = 0$, assuming a single batch.

Encoder. For each cell c, the encoder layer $f_E(\cdot)$ compresses the high-dimensional batch-corrected single-cell expression profile $x_c \in \mathbb{R}^N$ into a low-dimensional, latent representation $h_c \in \mathbb{R}^M, M < N$. The encoder layer is given by

$$h_c = f_E(x_c) = r(W_E x_c + b_E)$$

with learnable parameters $W_E \in \mathbb{R}^{M\times N}$ and $b_E \in \mathbb{R}^M$.

Decoder. The decoder layer $f_D(\cdot)$ decompresses the latent representation h_c to an output $y_c \in \mathbb{R}^N$ of the same dimension as the input single-cell profile, and is given by

$$y_c = f_D(h_c) = r(W_D h_c + b_D)$$

with learnable parameters $W_D \in \mathbb{R}^{N\times M}$ and $b_D \in \mathbb{R}^N$.

Imputer. We developed a self-correcting layer $f_I(\cdot)$ to impute missing entries based on y_c to an output v_c, inspired by a previously developed reconstruction approach[21]. In a two-step process, we first reduce the number of parameters by transforming the decoder layer output y_c to a p-dimensional latent vector by

$$u_c = r(W_U y_c + b_U) \in \mathbb{R}^p$$

with learnable parameters $W_U \in \mathbb{R}^{p\times N}$ and $b_U \in \mathbb{R}^p, p < N$ (we set $p = 64$). Then, we performed imputation by

$$v_c = P_{Z_c}[r(W_V u_c + b_V)] \in \mathbb{R}^N$$

with learnable parameters $W_V \in \mathbb{R}^{N\times p}$and $b_V \in \mathbb{R}^N$. Here, Z_c ($\overline{Z}_c$) is the set of zero-valued (non-zero-valued) genes in profile x_c of the cth cell, and the entry-sampling operator P_{Z_c} sets all entries not in Z_c to zero (that is, for vector r, with entries r_j, $P_{Z_c}(r_j) = r_j$ if $j \in Z_c$ and $P_{Z_c}(r_j) = 0, j \notin Z_c$). We used Z_c, as we are interested only in imputation for zero-valued genes in the profile x_c.

After obtaining the imputed vector v_c, we calculated the corrected single-cell expression profile: $\hat{x}_c = x_c + v_c$. This corrected single-cell profile $\hat{x}_c$ was then re-sent through the encoder-decoder framework to re-learn an updated latent representation.

Learning the objective function. The imputation layer imposes a recurrent structure on the scScope network architecture. For clarity of exposition, the recurrent scScope can be unfolded into multiple time steps (Supplementary Fig. 10 shows three steps). Then, the whole recurrent scScope framework can be described as

$$x_c = f_B(\tilde{x}_c), h_c^t = f_E(\hat{x}_c^t) = f_E(x_c + v_c^{t-1}),$$
$$y_c^t = f_D(h_c^t), v_c^t = f_I(y_c^t), v_c^0 = 0$$

for iterations $t = 1\ldots T$. At the first step, the correcting layer's output v_c^0 is set as zero. For $T = 1$, scScope is a standard autoencoder.

The learning objective for scScope is defined by the pursuit of unsupervised self-reconstruction (as typically used in autoencoder training):

$$f_B, f_E, f_D, f_I = \mathrm{argmin}\, L = \sum_{c=1}^{n}\sum_{t=1}^{T} \|P_{\overline{Z}_c}[y_c^t - x_c]\|^2$$

The entry-sampling operator $P_{\overline{Z}_c}$ forces loss computation only on non-zero entries of x_c. The parameters in the batch correction layer (f_B), encoder layer (f_E), decoder layer (f_D) and imputation layer (f_I) are all learned by minimizing the above loss function.

Our multiple GPU training strategy is outlined in Supplementary Fig. 1.

Cell subpopulation discovery. Representations outputted by scScope at each step can be concatenated as a long feature vector, which is easily integrated with any clustering method. We used the graph-based method PhenoGraph[20], as it performs automated robust discovery of subpopulations, as well as determines subpopulation numbers automatically.

Graph clustering for moderate-scale data. We directly applied the PhenoGraph software to datasets of moderate scale. All clustering results were obtained using a Python-implemented PhenoGraph package (v.1.5.2). We followed the suggested setting and considered 30 nearest neighbors when constructing graphs.

Scalable graph clustering for large-scale data. scScope enables the feature learning on millions of cells. However, PhenoGraph is unable to handle millions of cells because of the extreme computational costs (Supplementary Fig. 11) and memory requirements in graph construction. To leverage the power of graph clustering in analyzing these large-scale data, we designed a density downsampling clustering strategy by combining k-means and PhenoGraph.

In detail, we divided cells into M groups with equal size and performed k-means clustering on each group independently (Supplementary Fig. 6). The whole dataset was split into $M \times K$ clusters, and we only input the cluster centroids into PhenoGraph for graph clustering. Finally, each cell was assigned to graph clusters according to the cluster labels of its nearest centroids.

In our implementation on the dataset of 1.3 million mouse brain cells, we took $M = 500$ and $K = 400$, which made it possible to process millions of data in only tens of minutes without loss of accuracy.

Scalable memory allocation for analyzing large numbers of genes in large datasets. For large datasets and gene numbers, scScope implements a scalable memory allocation strategy that allows the dataset to be broken into a smaller number of batches that can be loaded directly into memory. We note that when this option is used, minibatches are only selected from in each batch during training. This option was only used in Fig. 2c for the case of ≥10,000 genes; here the 400,000 mouse cell atlas dataset was broken into four batches each of size 100,000.

Methods compared. All compared methods were run on the same server with Xeon E5 central processing unit, 64 GB memory, Nvidia Titan X GPU and Ubuntu 14.04 operation system. Further, all comparisons were performed using log-transformed input (we observed similar relative performance of the six compared methods above across four different choices of input scaling or normalization methods; see Supplementary Fig. 12).

Unless otherwise noted, software packages were used with their default values. We observed that all methods were reasonably robust to changes in the latent dimension M (for example, Supplementary Fig. 13), and to avoid an intractable number of possible comparisons, we set $M = 50$ for all comparisons. For all deep-learning methods (autoencoder, DCA, scVI and scScope), we set the learning rate to 10^{-3}, batch size to 64 and epoch = 100.

PCA. PCA is implemented using the Python package scikit-learn v.0.18.

Autoencoder. The simple autoencoder is implemented using scScope with $T = 1$. Variations of the autoencoder are described in the legend of Supplementary Fig. 14.

MAGIC. The MAGIC algorithm was performed using the Python-based package magic[11]. We inputted the raw data and employed the library_size_normalize() function provided by the software for all simulated and real data.

ZINB-WaVE. We employed the R package zinbwave[8].

Single-cell interpretation via multi-kernel learning (SIMLR). We used the Python implementation of SIMLR[9]. SIMLR needs to take the desired cluster number as input. For our simulated dataset, we input the known cluster numbers. For the retina dataset, where the 'true' cluster number is unknown, we set it to 39, which is the cluster number reported in the original study[4].

DCA. We used the TensorFlow-based Python package of DCA[15] (download date: 4 September 2018).

Single-cell variational inference (scVI). We used the Torch-based Python package of scVI[14] (download date: 5 June 2018).

scScope. scScope was implemented in Python v.3.6 with the TensorFlow-GPU v.1.4.1, Numpy v.1.13.0, Scikit-learn v.0.18.1 packages. Unless noted otherwise, for comparisons of methods, scScope was run with default setting (for example, single GPU mode); hyperparameter choices are given in Supplementary Table 2, and the results of varying them are demonstrated in Supplementary Fig. 15.

Downsampling training strategies on a large-scale dataset. Some non-deep-learning-based approaches could not run directly on extremely large datasets. For these approaches, we randomly downsampled datasets to 20,000-cell subsets. On these downsampled datasets, single-cell feature vectors were learned by the respective method and clustered by PhenoGraph. A support vector machine was trained on this subset in the latent feature space and then used to assign labels for the rest of cells in the unsampled dataset. The deep-learning approaches we tested could learn features on millions of cell profiles, but the software packages did not provide a function for automatic clustering. For comparisons, we passed the output of their learned single-cell features to the scalable clustering approach.

Evaluation of clustering and batch correction. *ARI.* We used the adjusted Rand index (ARI)[22,23] to compare label sets of two clustering methods. For two clustering results U and V with r and s clusters on a dataset of n cells, n_{ij} denotes the number of cells shared between cluster i in U and cluster j in V. ARI is defined as

$$\mathrm{ARI}=\frac{\sum_{ij}\binom{n_{ij}}{2}-\left[\sum_i\binom{n_{i\star}}{2}\sum_j\binom{n_{\star j}}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_{i\star}}{2}+\sum_j\binom{n_{\star j}}{2}\right]-\left[\sum_i\binom{n_{i\star}}{2}\sum_j\binom{n_{\star j}}{2}\right]/\binom{n}{2}}$$

where $n_{i\star}=\sum_j n_{ij}$, $n_{\star j}=\sum_i n_{ij}$ and n is the number of cells in the dataset.

Entropy of batch mixing. To evaluate the performance of batch correction, we made use of the score developed in mutual nearest neighbors[24]. In brief, 100 cells were randomly sampled from the entire population, the entropy of the distributions of batches for the nearest 100 neighbors was calculated and the process was iterated 100 times, with results shown as box plots.

Evaluation of imputation. *Imputation error for simulated data.* On simulated data, the ground truth dropout vector l is known. The imputation accuracy was defined as the normalized distance between the imputed log count entries and log count ground truth entries. We constructed lists l and $\hat{l}$ whose elements correspond to either ground truth or imputed values (respectively) for all dropout entries across all cells (Supplementary Fig. 2). We defined the normalized error as

$$\mathrm{Error}=\frac{\|l-\hat{l}\|_1}{\|l\|_0}$$

where $\|\,\|_p$ means the l^p norm of a vector.

Reconstruction error on held-out biological data. For real biological data, the ground truth values for missing genes are unknown. To evaluate scScope's imputation accuracy on real biological data, we followed the same downsampling strategy as used for scVI[14]. Namely, we randomly split the entire collection of n cells into n_{train} training cells and n_{val} validation cells. We used the different imputation methods to build gene models from the n_{train} cells. On each of the n_{val} cells, we randomly set p% of its non-zero genes as 'simulated' missing genes and set their corresponding count values to zero. The real measured values of these simulated missing genes were then used to generate the ground truth list l, and the list $\hat{l}$ was based on inferred values for the simulated missing genes from the n_{val} cells. The reconstruction error was calculated as for simulated data above.

Simulated datasets. *Simulation with Splatter.* The simulation package Splatter[16] is designed to generate realistic scRNA-seq data. We used this package to generate data with 2,000 cells, three subpopulation groups and dropout rates from one to five.

Simulation with SIMLR. SIMLR[9] was used to generate large-scale scRNA-seq data because of limitations in the scalability of Splatter. High-dimensional single-cell expression data $x_c\in\mathbb{R}^N$ are generated by a latent code $z_c\in\mathbb{R}^P\,(P<N)$, which is not observable. z_c is sampled from a Gaussian mixture model with k Gaussian components, that is, $z_c\approx\sum_{j=1}^{k}\pi_j N(\mu_j,\Sigma_j)$. The mixture coefficients π_j were chosen from a uniform distribution and normalized to sum to 1, the mean vector $\mu_j\in\mathbb{R}^P$ was uniformly sampled in $[0,1]^P$, and the covariance matrix $\Sigma_j\in\mathbb{R}^{P\times P}$ was chosen to be $\Sigma=0.1\times I$, for identity matrix I.

To simulate single-cell gene vectors, we generated a projection matrix $A\in\mathbb{R}^{N\times P}$ to map the low-dimensional latent code to a high-dimensional space. First, we simulated ground truth, x_c^{true}, which cannot be observed:

$$x_c^{\mathrm{true}}=Az_c+b$$

where each entry in A was independently sampled from the uniform distribution $U(-0.5,0.5)$ and bias $b=0.5$ was added to avoid negative gene expression in the high-dimensional mapping. Second, we simulated the observed profile, x_c^{obs}, which may contain gene-specific noise and dropout artifacts due to the sequencing technique and platform. Noise was added to the true gene profile by $x_{cg}^{\mathrm{noise}}=x_{cg}^{\mathrm{true}}+\varepsilon_{cg}$, where $\varepsilon_{cg}\sim N(0,\Sigma_g^{\mathrm{noise}})$ and $\Sigma_g^{\mathrm{noise}}$ was uniformly sampled in the range of [0, 0.2] independently for each gene. Dropout events were added via a double exponential model with decay parameter[25] α:

$$x_{cg}^{\mathrm{obs}}=x_{cg}^{\mathrm{noise}}\delta\left[q_{cg}>\exp{-\alpha\ x_{cg}^{\mathrm{noise2}}}\right]$$

where x_{cg} denotes the gth gene of x_c, q_{cg} was randomly sampled in [0, 1], and $\delta=1$ if its argument is true and 0 otherwise.

Simulation with rare cell subgroups. To generate cell subpopulations with rare cell types, we sampled the mixture coefficients π_j from a non-uniform distribution as

$$\pi_{1\ldots k_m}=q;\pi_{k_m+1\ldots k}=\frac{1-q*k_m}{k-k_m}$$

where $q\ll 1/k$ is the mixture fraction for each minor cluster, k_m is the number of rare cell subpopulations, and k is the total number of subpopulations.

Biological datasets. *Lung tissue data.* The lung tissue dataset is part of the Mouse Cell Atlas data[6]. This dataset was downloaded from Gene Expression Omnibus (GEO) database (accession number GSE108097). In this dataset, 6,940 cells were sequenced via three independent experiments by Microwell-seq, with 2,512, 1,414 and 3,014 cells in each batch. In total, 1,000 highly variable genes were selected for analysis.

CBMC dataset. In this dataset, 8,617 CBMCs were profiled by CITE-seq, a new technology that enabled the simultaneous measurement of protein levels and transcriptome levels for each cell[26]. This dataset was downloaded from the GEO database (accession number GSE100866). Cell types in CBMC have been extensively studied and identified. On the basis of this previous knowledge, 13 monoclonal antibodies were chosen to identify bone fide cell types. These antibodies serve as an 'orthogonal' ground truth to evaluate analysis results based on RNA-seq data.

In the data pre-processing stage, we removed the spiked-in mouse cells and only kept 8,005 human cells for analysis using the cell-filtering strategy introduced in the original study[26]. The top 1,000 most variable human genes were selected for downstream analysis after the log(1 + x) transformation and normalization by library size. For antibody data, we used the centered log ratio transformed antibody-derived tags (ADTs), which is also provided by authors.

To evaluate the performance of each method, we first automatically identified cell populations on the basis of ADT data using PhenoGraph. Then, scRNA-seq data were input to each method to learn latent representations that were used by PhenoGraph to predict cell types. The ADT-derived cell types were taken as ground truth to evaluate the accuracy of cell types by scRNA-seq data.

Retina data. In this dataset, 44,808 cells were captured from the retinas of 14-d-old mice and sequenced by DropSeq[4]. Data were obtained from the GEO database (accession number GSE63473). We followed previous procedures to select the 384 most variable genes[4] and then perform the log(1 + x) transformation on their expression values. After clustering, we annotated clusters using the same marker genes as in the original study[4].

We identified candidate cell types on the basis of the highest average type-specific marker expression (Supplementary Table 4). For each cluster, we calculated the fold-change values of all cell-type markers, and if at least one type-specific gene marker was expressed significantly higher ($\log_2$ fold change > 0.5) than in all other clusters, we assigned the cluster with the candidate cell type. Otherwise the cluster was assigned to the cell type 'Rod cell'.

Mouse cell atlas data. The mouse cell atlas dataset is designed to offer a comprehensive investigation of all main cell types in mouse[6]. Data were downloaded from the GEO database (accession number GSE108097). In the dataset, 405,796 cells were sampled from 51 tissues and were sequenced by Microwell-seq.

Data were first normalized by library size, and 1,000, 2,000, 5,000, 10,000 or 20,000 top-variable genes were selected to test the scalability of each method on gene numbers. Only the deep-learning-based single-cell tools (DCA, scVI and scScope) could be applied directly to this large-scale dataset. Further, to identify clusters in the mouse cell atlas dataset, we applied our scalable clustering approach to the latent features.

In most of the 51 tissues, one main cell type dominated the cell population (see Fig. 2b,c in ref. [6]). Therefore, we used the tissue identity as a proxy for ground truth to evaluate cell-type discovery.

Intestinal data. In this dataset, intestinal epithelial cells were captured from ten mice and sequenced using droplet-based scRNA-seq[27]. Data were downloaded from the GEO database (accession number GSE92332). Among all cells, 1,770 cells from two mice were infected by *Salmonella* for 2 d; 2,121 cells (2 mice) and 2,711 cells (2 mice) were infected by *H. polygyrus* for 3 and 10 d, respectively. An additional 3,240 cells were sequenced from four healthy mice as a control group.

We again followed the same procedure that log-transformed the expression data and selected top 1,000 most variable genes as input to scScope. For cell subpopulation annotation, we first assigned clusters to one of seven main cell types (stem, cell-cycle related, distal enterocyte, proximal enterocyte, goblet, enteroendocrine and tuft) according to the maximum averaged expression of cell-type makers (Supplementary Table 10). Second, cell-cycle-related clusters were subdivided into increasing stages of maturation (transit amplifying early stage, transit amplifying late stage and enterocyte progenitor) on the basis of the ratio of cell-cycle and stem cell markers to enterocyte expression (Supplementary Table 11). Third, the distal and proximal enterocyte clusters were further classified (immature versus mature) on the basis of increasing expression levels of the enterocyte gene markers.

After annotating clusters, we calculated the cell proportion for each mouse and then averaged the proportions among mice of the same infection condition. For significant tests of proportion changes after infection, we compared proportions of mice in the control group and in the infection group using a two-sided *t*-test and rank-sum test. *P* values were obtained under the null hypothesis that no changes happened in proportions after infection.

Overexpressed genes for each cluster were also identified by the same differential expression analysis.

Brain data. Data were obtained from 10X Genomics (http://10xgenomics.com). 1,308,421 cells from embryonic mouse brains were sequenced by Cell Ranger 1.3.0 protocol. We transformed unique molecular identifier count data into log(TPM + 1) (ref. [4]) and calculated the dispersion measure (variance/mean) for each gene. According to the rank of the dispersion measure, we selected the top 1,000 most variable genes for analysis.

For fast exploration of the data in Fig. 3, where no comparisons to other methods were used, we used a batch size of scScope to 512, trained the model for ten epochs and used four GPUs. Cells were further clustered into 36 groups by our density downsampling clustering. We annotated clusters to three main types (excitatory neurons, inhibitory neurons and non-neuronal cells) on the basis of maximally expressed marker genes (Supplementary Table 7).

To identify cluster-specific overexpressed genes, we then conducted differential expression analysis for each gene. We normalized the unique molecular identifier count to the range of [0 1] for each gene, enabling comparisons across genes. Then gene-expression fold-change and rank-sum *P* values were calculated between cells in versus outside each cluster. Significantly overexpressed genes were identified using the criteria of $\log_2$ fold change > 0.5 and rank-sum $P < 0.05$.

The dataset was downloaded from https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.3.0/1M_neurons on 10 December 2017.

The data analysis by 10X Genomics was obtained from http://storage.pardot.com/172142/31729/LIT000015_Chromium_Million_Brain_Cells_Application_Note_Digital_RevA.pdf.

Software used in the study. We used the following software: MAGIC, https://github.com/KrishnaswamyLab/MAGIC; ZINB-WaVE, https://github.com/drisso/zinbwave; SIMLR, https://github.com/bowang87/SIMLR_PY; DCA, https://github.com/theislab/dca; scVI, https://github.com/YosefLab/scVI; PhenoGraph, https://github.com/jacoblevine/PhenoGraph; and Splatter, https://github.com/Oshlack/splatter-paper.

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Code availability

scScope can be obtained as an installable Python package, via 'pip install scScope', and is available under the Apache license. All software, instructions and software updates will be maintained at https://github.com/AltschulerWu-Lab/scScope.

Data availability

Data for this paper are available in its text and Supplementary Information files or from the websites or database accessions referenced in the paper.

References

21. Franke, L. et al. *Am. J. Hum. Genet.* **78**, 1011–1025 (2006).
22. Hubert, L. & Arabie, P. *J. Classif.* **2**, 193–218 (1985).
23. Rand, W. M. *J. Am. Stat. Assoc.* **66**, 846–850 (1971).
24. Haghverdi, L., Lun, A. T. L., Morgan, M. D. & Marioni, J. C. *Nat. Biotechnol.* **36**, 421–427 (2018).
25. Kharchenko, P. V., Silberstein, L. & Scadden, D. T. *Nat. Methods* **11**, 740–742 (2014).
26. Stoeckius, M. et al. *Nat. Methods* **14**, 865–868 (2017).
27. Haber, A. L. et al. *Nature* **551**, 333–339 (2017).

nature research

Corresponding author(s):	Steven Altschuler
Last updated by author(s):	Jan 24, 2019

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see Authors & Referees and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☒	☐	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	No software was used for data collection.
Data analysis	Splatter v1.4.3, MAGIC v1.3.0, SIMLR (https://github.com/bowang87/SIMLR_PY, no version number avaliable), ZINB-WaVE v1.4.0, DCA v0.2.2 and scVI v0.1.0 were used as comparing methods. Links to comparing methods were provided in the manuscript. scScope was implemented in Python 3.6 with TensorFlow-GPU 1.4.1, Numpy 1.13.0, Scikit-learn 0.18.1 packages. Scikit-learn 0.18.1 Python package was used for PCA , KMeans clustering, Pearson correlation and t-SNE visualization. PhenoGraph v1.5.2 was used for graph-based clustering.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

Six published datasets were used in this study. Mouse cell atlas (including lung tissue dataset), cord blood mononuclear cells, retina and gut datasets were obtained from Gene Expression Omnibus (GEO) database (accession number: GSE108097, GSE100866, GSE63473 and GSE92332). Mouse brain data were download from https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.3.0/1M_neurons on December 10, 2017. All datasets can be downloaded with respect to the website policy.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

[✓ Life sciences] ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	No experiments in study.
Data exclusions	In the cord blood mononuclear cells dataset, we removed the spiked-in mouse cells and only kept 8,005 human cells for analysis using the cell-filtering strategy introduced in original study.
Replication	No experiments in study.
Randomization	No experiments in study.
Blinding	No experiments in study.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study	
☒	☐	Antibodies
☒	☐	Eukaryotic cell lines
☒	☐	Palaeontology
☒	☐	Animals and other organisms
☒	☐	Human research participants
☒	☐	Clinical data

Methods

n/a	Involved in the study	
☒	☐	ChIP-seq
☒	☐	Flow cytometry
☒	☐	MRI-based neuroimaging