Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning

Supplementary Information

Bo Wang[a], Junjie Zhu[b], Emma Pierson[a], Daniele Ramazzotti[a,c], Serafim Batzoglou[a]

[a] *Department of Computer Science, Stanford University, Stanford, CA, USA*
[b] *Department of Electrical Engineering, Stanford University, Stanford, CA, USA*
[c] *Department of Pathology, Stanford University, Stanford, CA, USA*

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Figures

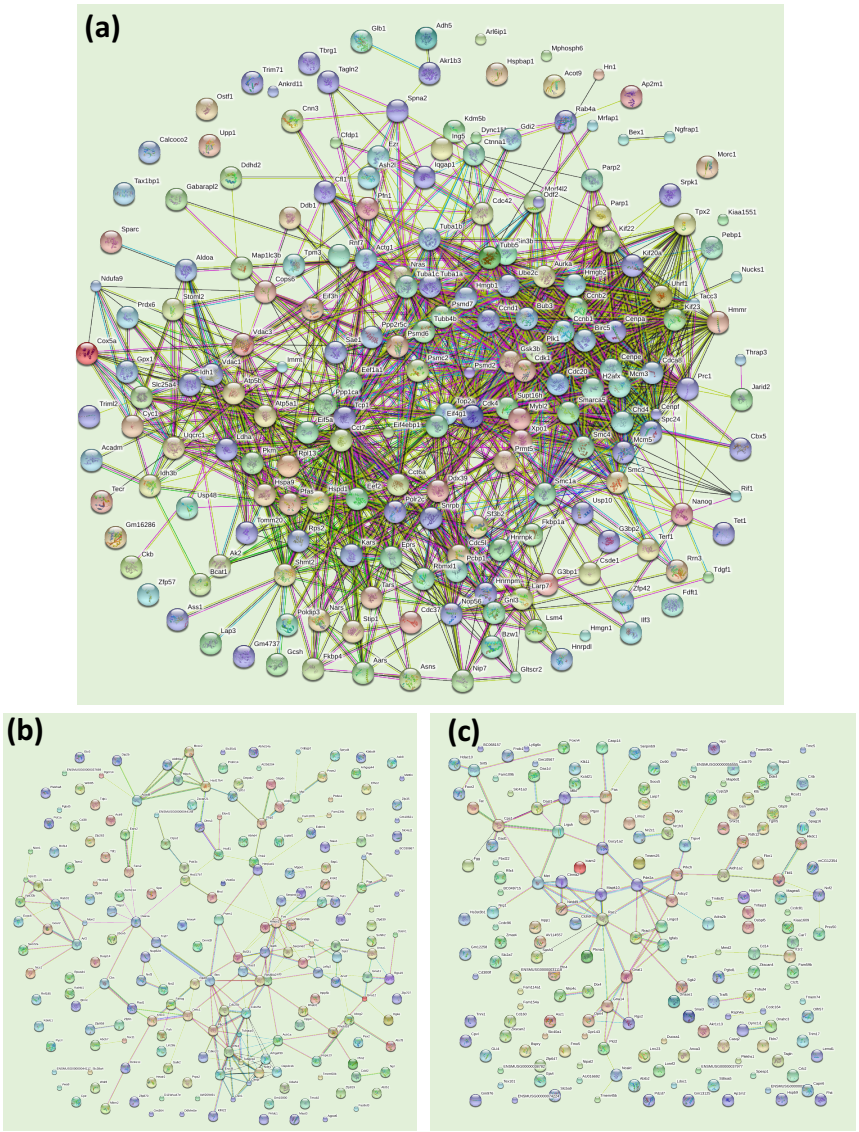

Figure 1: Protein-protein interaction networks from the STRING database [1]. (a) Using the top 200 genes by SIMLR's gene prioritization. (b) Using the top 200 genes by variance ranking. (c) Using the top 200 genes by dispersion (variance over mean) ranking. Each node represents a gene's protein product and each edge represents the interaction or co-expression between the proteins. Because SIMLR is able to reveal gene sets that are consistent with the learned global similarity dominated by the cell-cycle effect, it is able to prioritize cell-cycle genes that are typically co-expressed.

Nature Methods: doi:10.1038/nmeth.4207

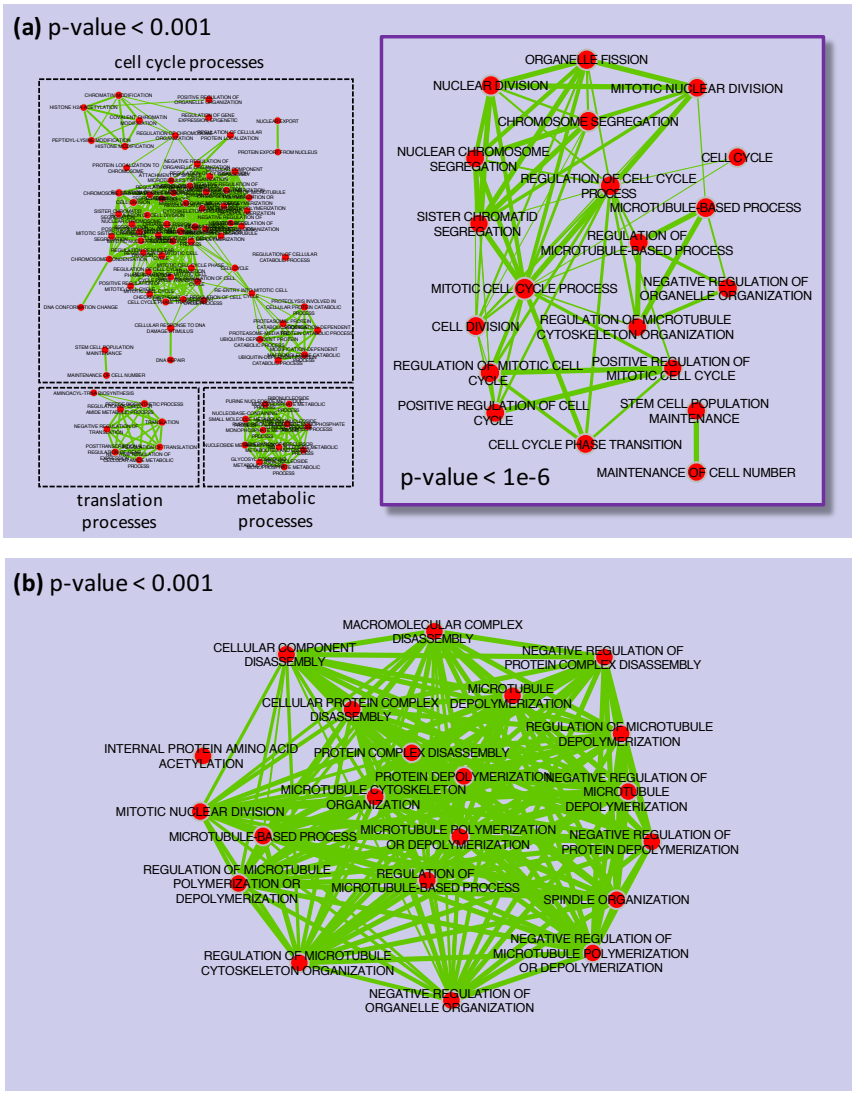

Figure 2: GO Enrichment Mapping the cluster-specific genes for the Buettner [2] data set. (a) The GO terms that are significantly associated with G2M-specific genes that are up-regulated. (b) The GO terms that are significantly associated with G1-specific genes that are jointly down-regulated. We first took the cell-state-specific gene sets (345 up-regulated genes for G2M and 145 down-regulated genes for G1 respectively) to identify significant GO terms using default parameters on the g:profiler web interface [3]. Then we used the enrichment map plug-in in Cytoscape [4] to visualize these significant GO terms. Each node represents a GO term and each edge represents the degree of gene overlap (Jaccard similarity) that exists between two gene sets corresponding to the two GO terms.

Nature Methods: doi:10.1038/nmeth.4207

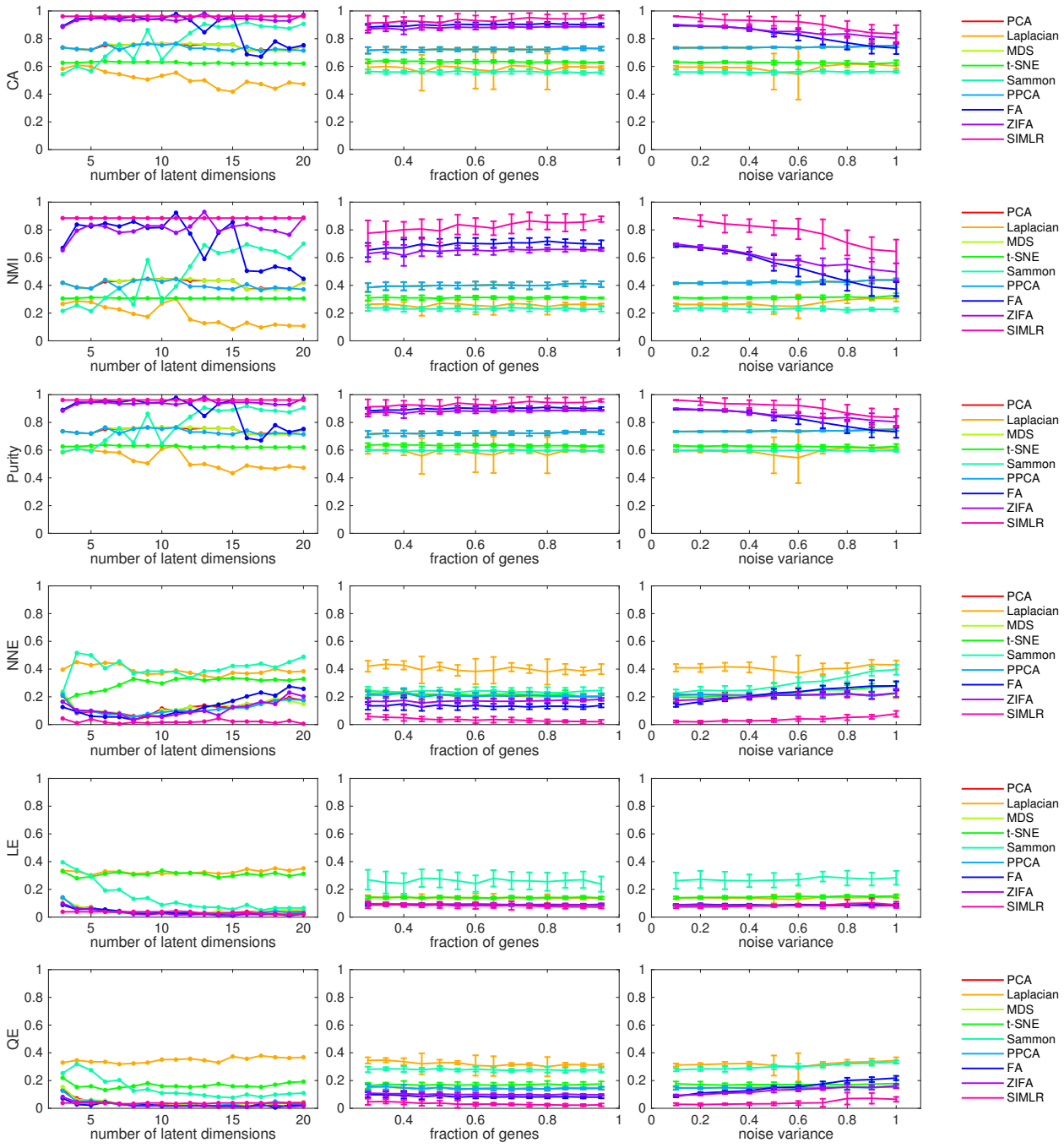

Figure 3: Sensitivity analysis of nine methods on the Buettner data set [2]. Each row of plots corresponds to a clustering/ separability metric. Each column corresponds to the sensitivity of each method to (1) the number of the latent dimensions selected, (2) the fraction of randomly selected genes, and (3) the variance of additive noise.

Nature Methods: doi:10.1038/nmeth.4207

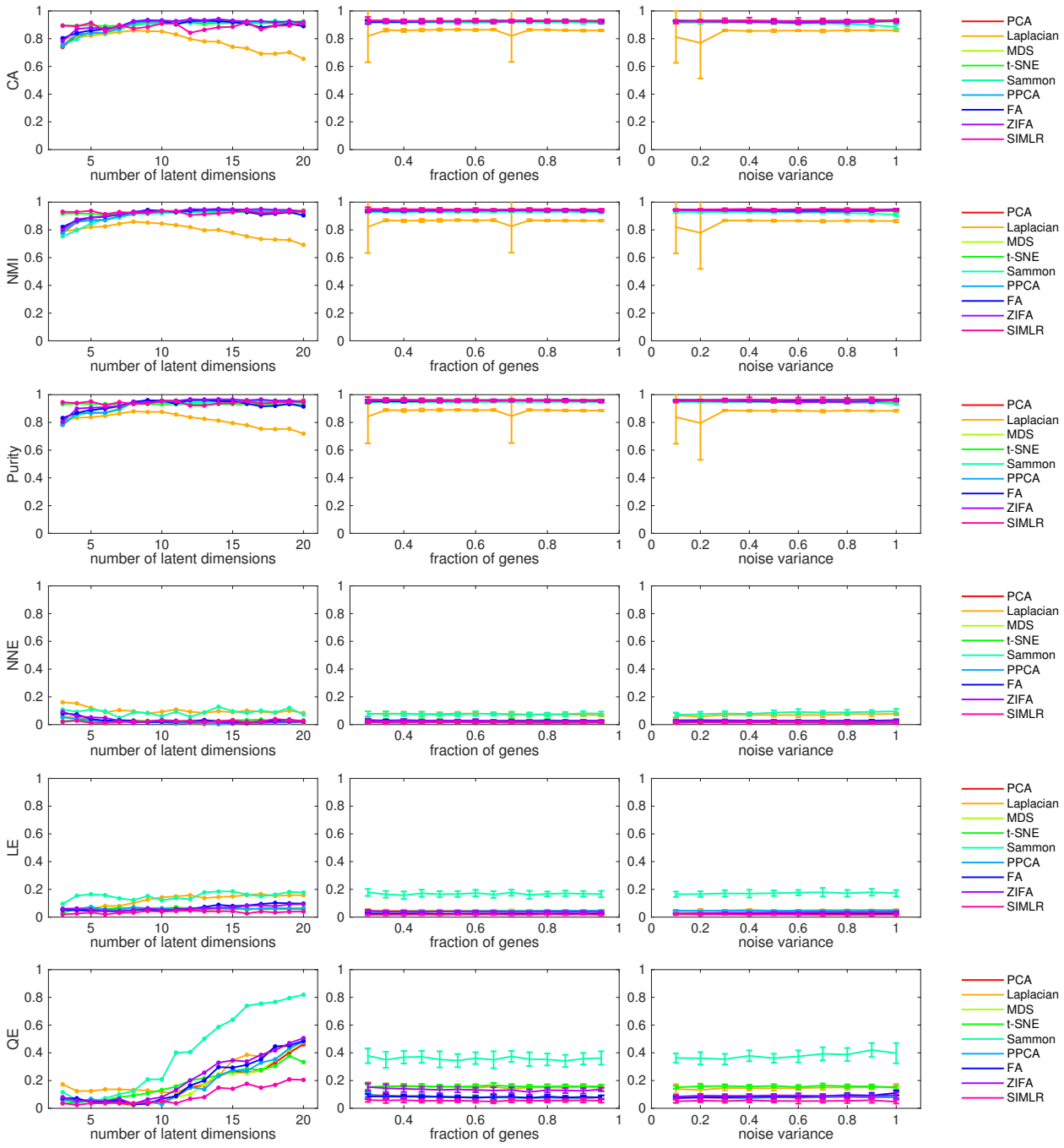

Figure 4: Sensitivity analysis of nine methods on the Pollen data set [5]. Each row of plots corresponds to a clustering/ separability metric. Each column corresponds to the sensitivity of each method to (1) the number of the latent dimensions selected, (2) the fraction of randomly selected genes, and (3) the variance of additive noise.

Nature Methods: doi:10.1038/nmeth.4207

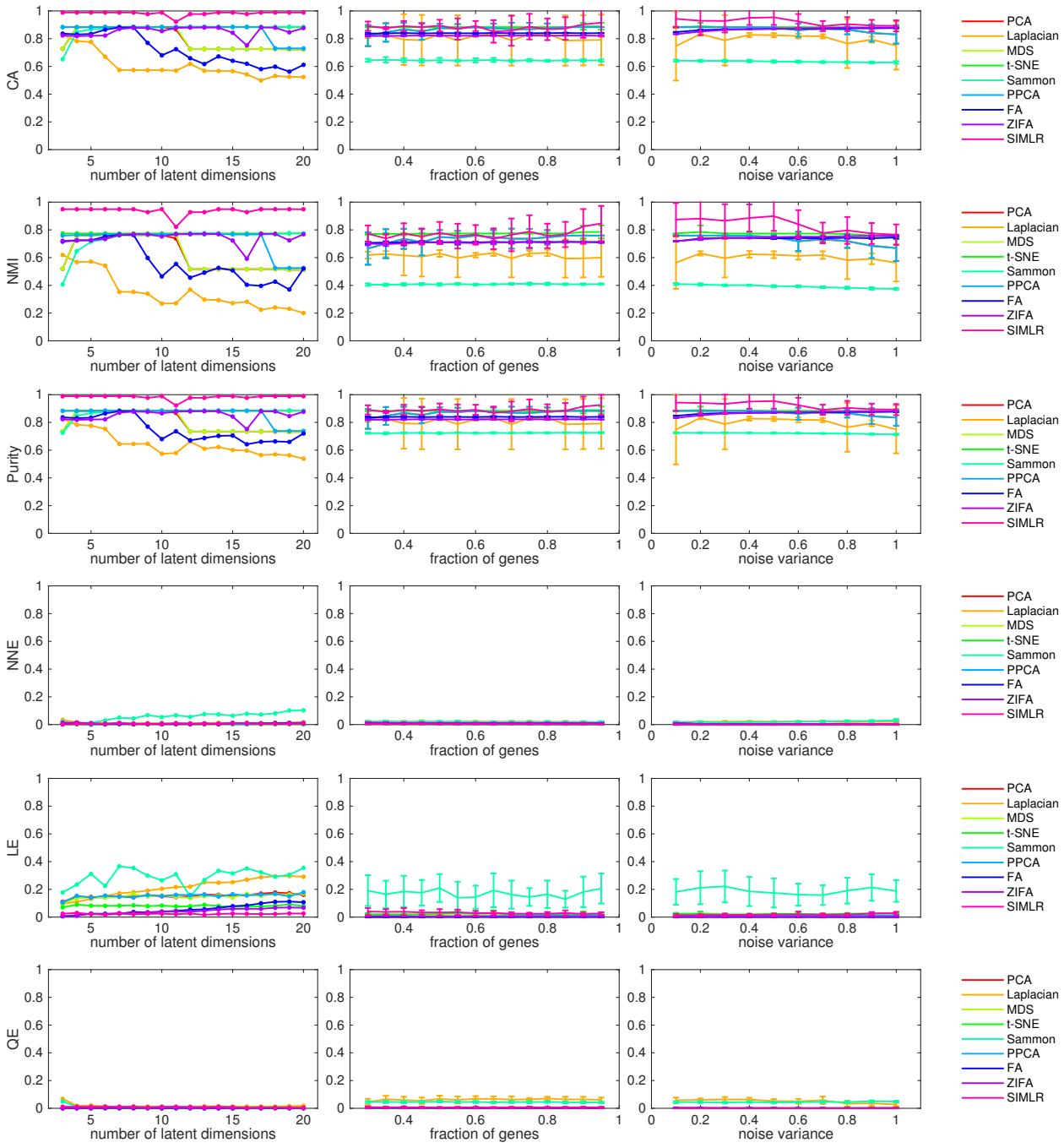

Figure 5: Sensitivity analysis of nine methods on the Kolodziejczyk data set [6]. Each row of plots corresponds to a clustering/ separability metric. Each column corresponds to the sensitivity of each method to (1) the number of the latent dimensions selected, (2) the fraction of randomly selected genes, and (3) the variance of additive noise.

Nature Methods: doi:10.1038/nmeth.4207

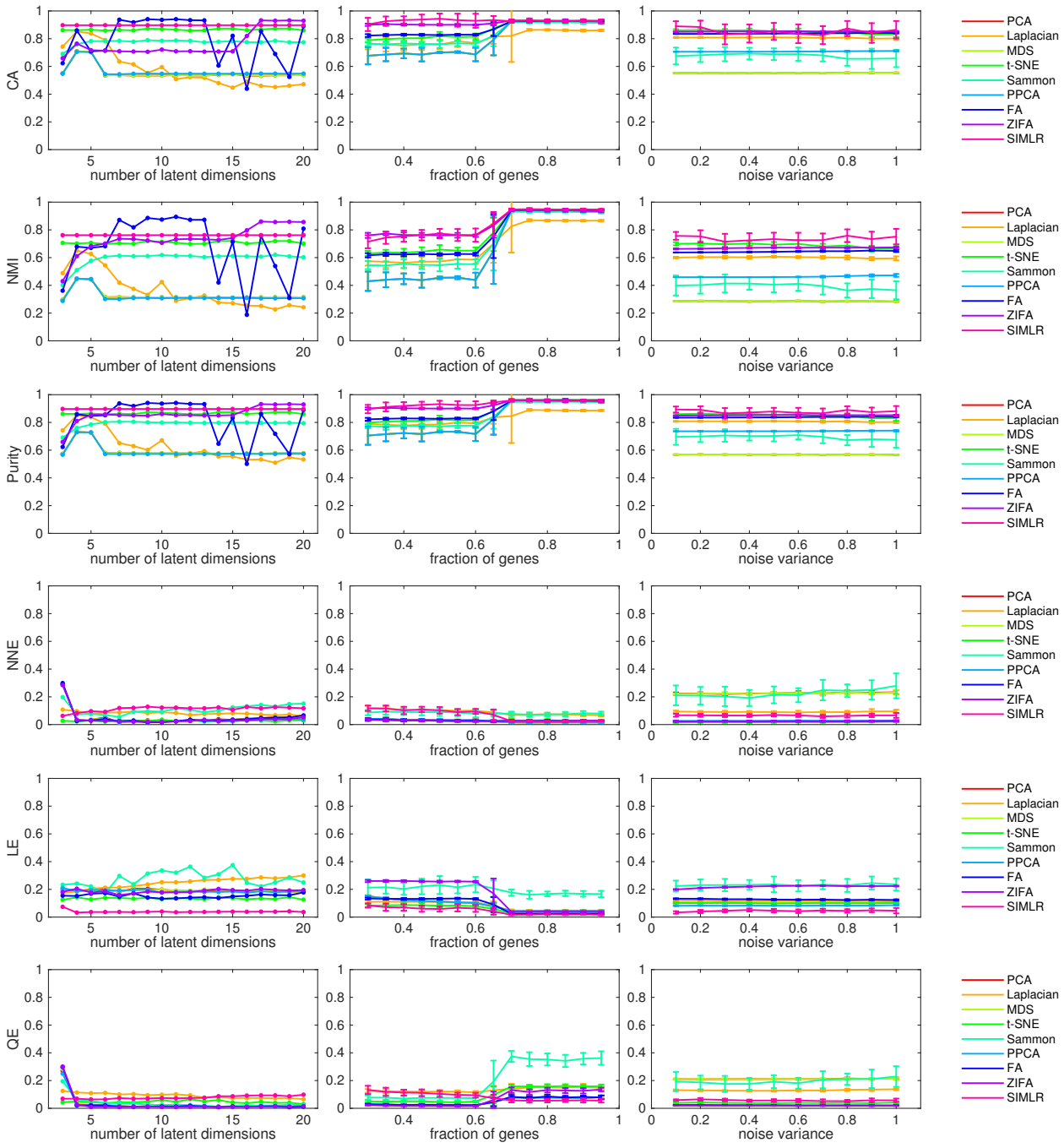

Figure 6: Sensitivity analysis of nine methods on the Usoskin data set [7]. Each row of plots corresponds to a clustering/ separability metric. Each column corresponds to the sensitivity of each method to (1) the number of the latent dimensions selected, (2) the fraction of randomly selected genes, and (3) the variance of additive noise.

Nature Methods: doi:10.1038/nmeth.4207

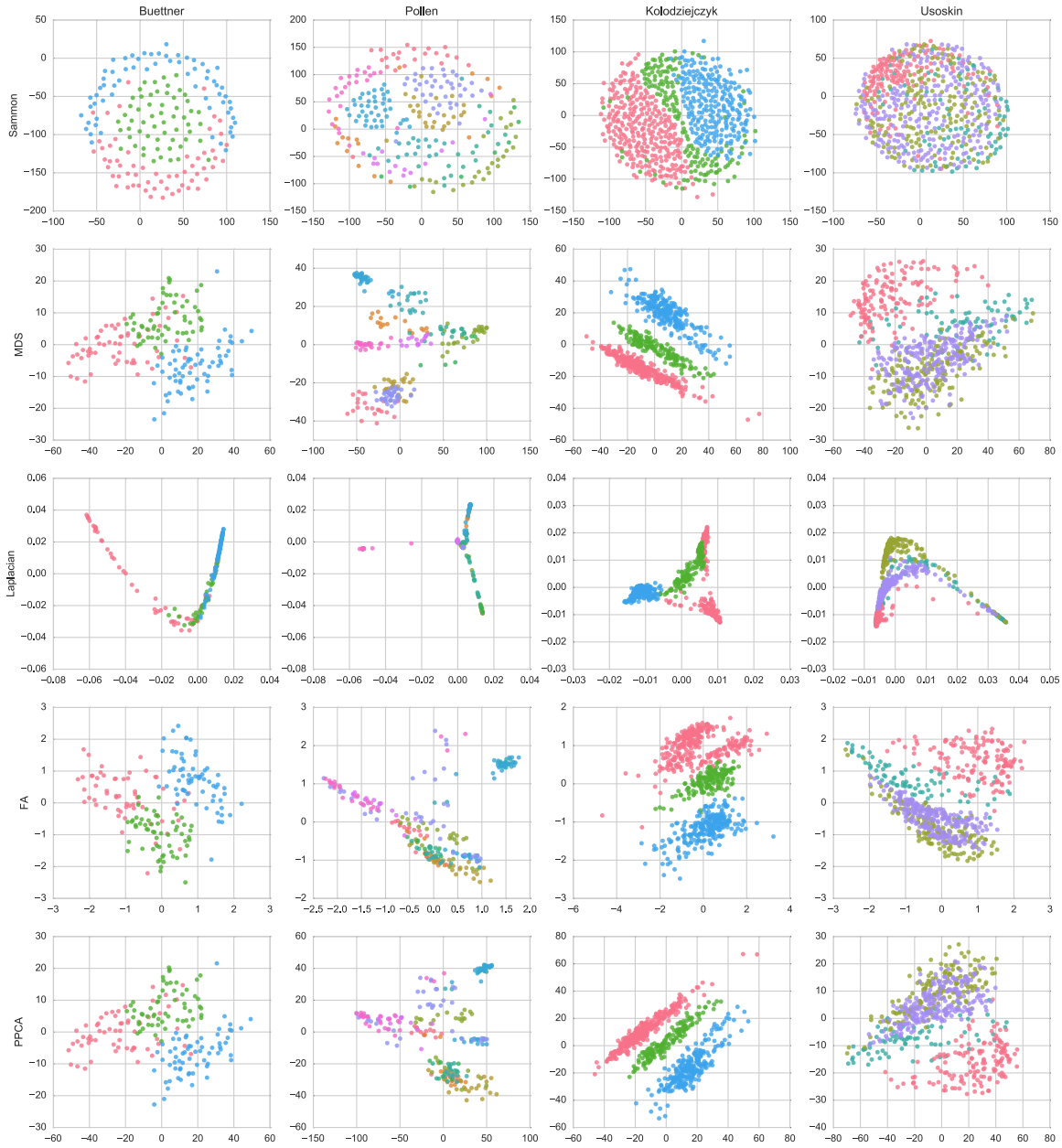

Figure 7: Scatter plots of four single cell data sets: the Buettner data set (3 populations) [2], the Pollen data set (11 populations) [5], the Kolodziejczyk data set (3 populations) [6], and the Usoskin data set (4 populations) [7]. Different colors indicate the true labels for different cell populations. Each data set was reduced to 2-D via five dimension reduction methods: Sammon [8], MDS [9], Laplacian [10], FA [11] and PPCA [12].

Nature Methods: doi:10.1038/nmeth.4207

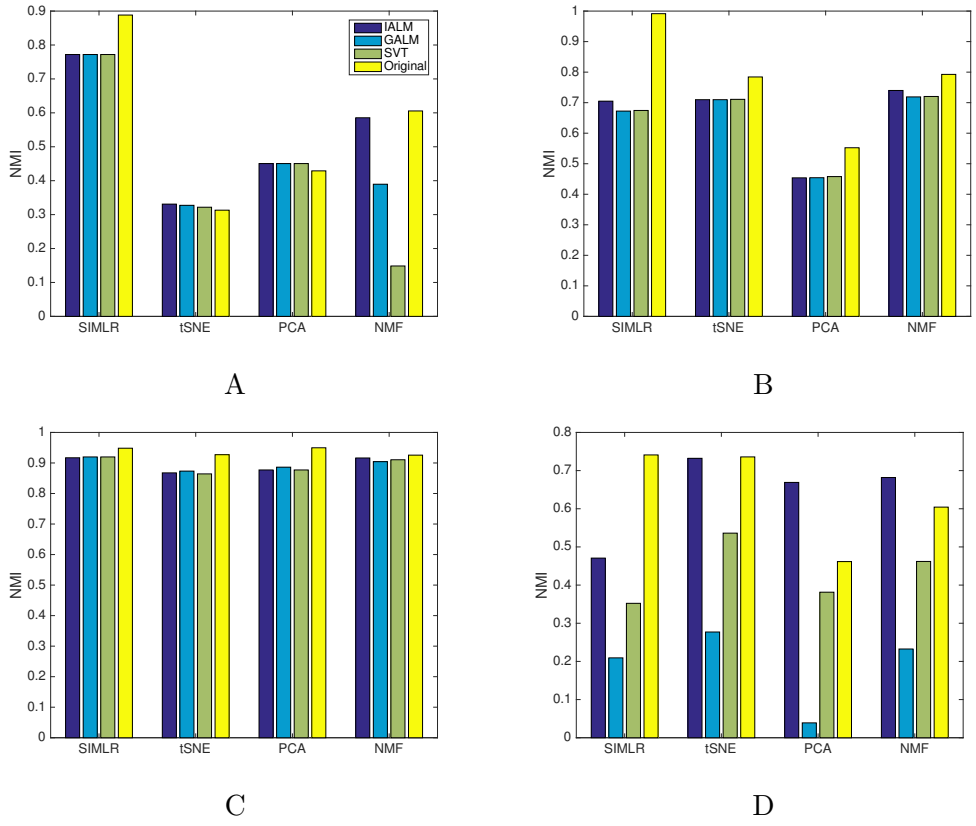

Figure 8: Effects on three state-of-arts matrix completion methods on the four real single-cell data sets: (A) Buettner [2], (B) Pollen [5], (C) Kolodziejczyk [6], and (D) Usoskin [7].

Nature Methods: doi:10.1038/nmeth.4207

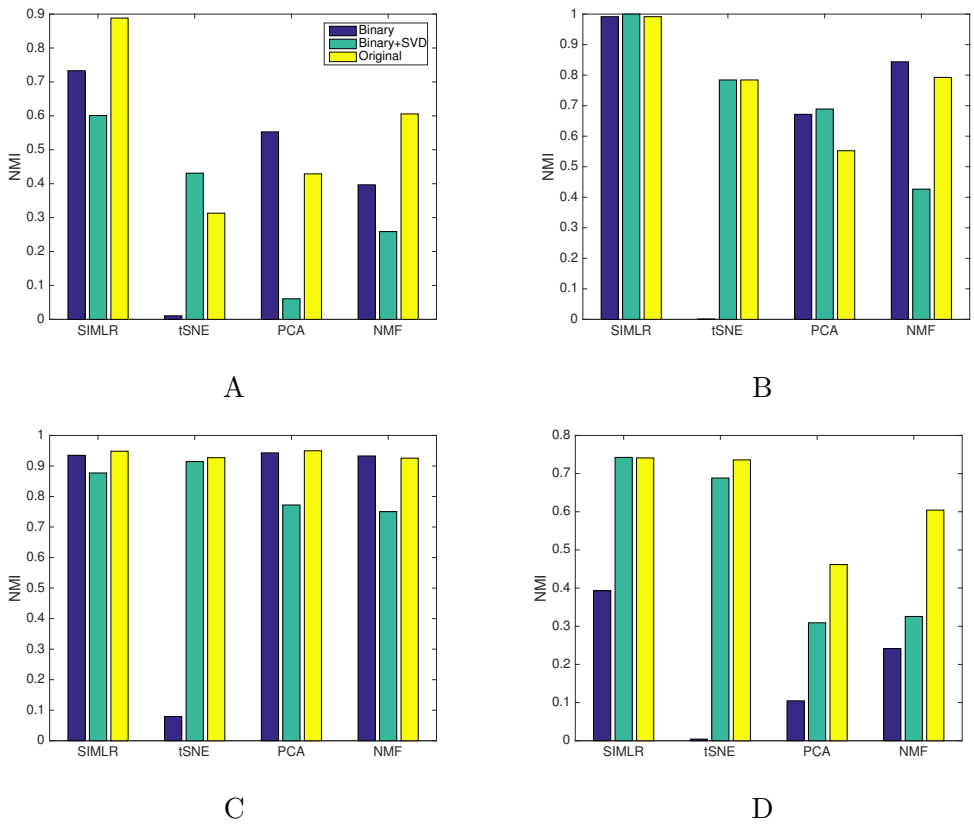

Figure 9: Effects on binarization on the four real single-cell data sets: (A) Buettner [2], (B) Pollen [5], (C) Kolodziejczyk [6], and (D) Usoskin [7].

Nature Methods: doi:10.1038/nmeth.4207

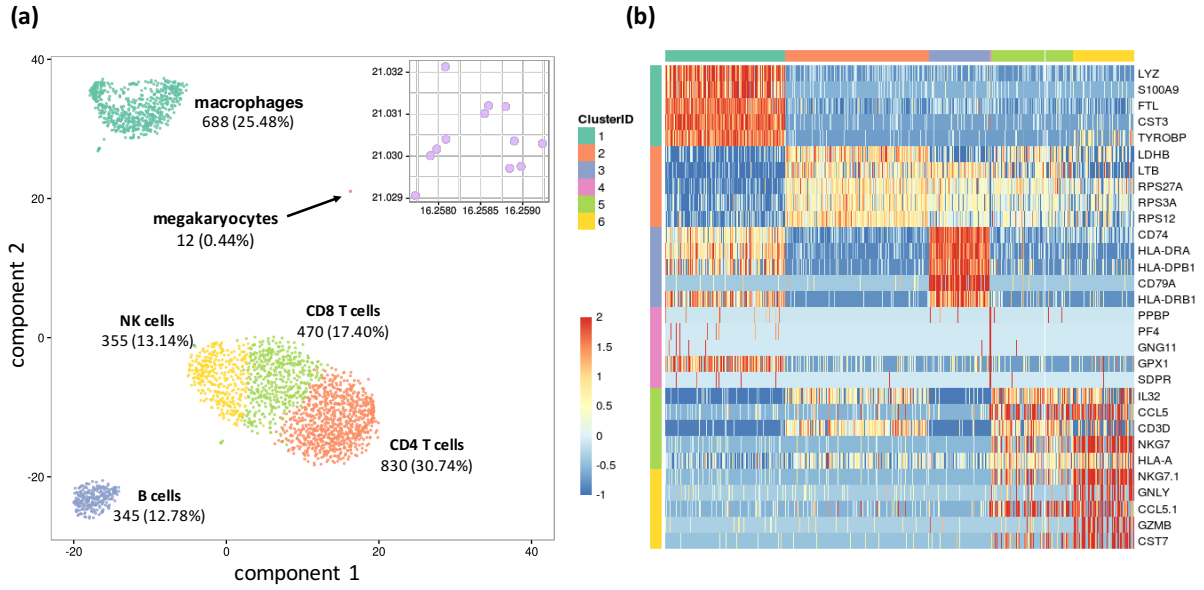

Figure 10: Unbiased analysis to identify cell types in human PBMCs. (a) A scatter plot using SIMLR's 2-D embedding with SIMLR's k-means cluster assignment (with $k = 5$). Each point represents a cell in the 2-D embedded space and colors represent the k-means cluster labels. The cell types are assigned by specific gene markers after clustering. (b) A heatmap of log10-scale expression values indicating top 5 most highly regulated genes corresponding to each cluster. Each column represents a cell and each row is a specific gene. The colored axis indicate the cluster assignments.

Nature Methods: doi:10.1038/nmeth.4207

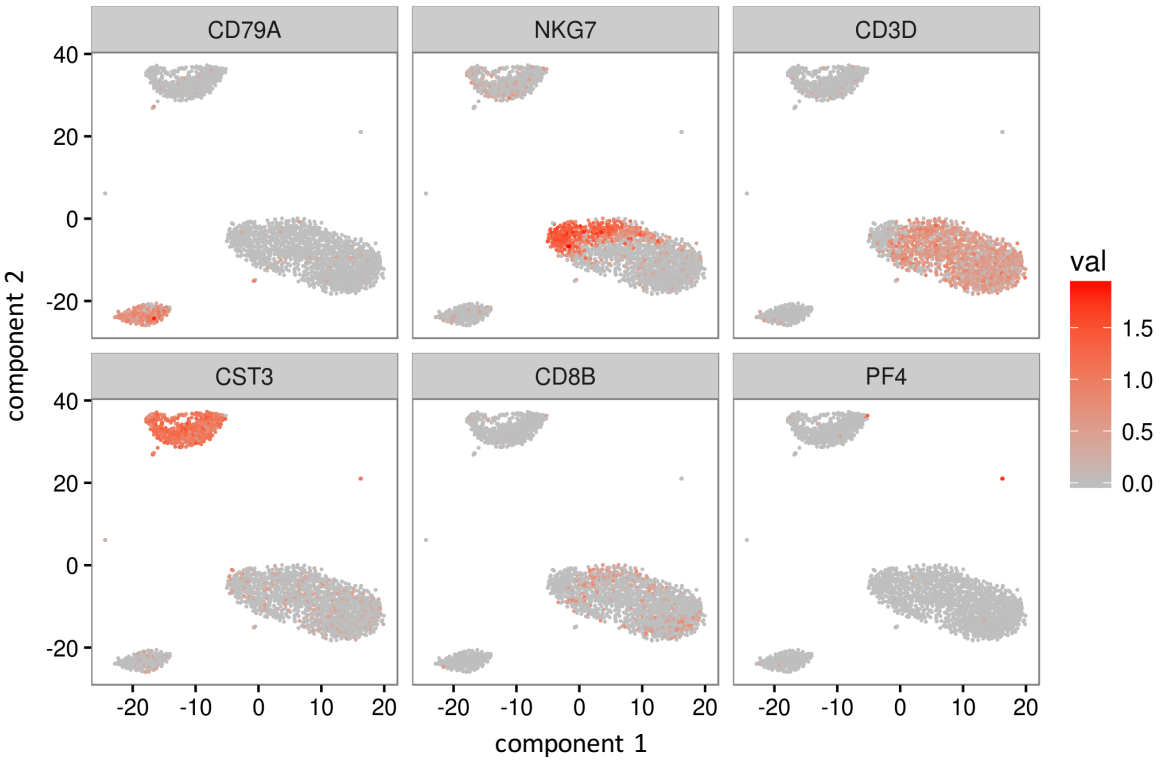

Figure 11: The log10-scale expression values of genes that are known markers for cell types in the human PBMC (e.g., CD79A is a B-cell marker; NKG7 is a marker for NK cells and also expressed in activated CD8+ T cells; CD3D is a T-cell marker; CST3 is a macrophage marker; CD8B is a CD8+ T cell marker; and PF4 is a megakaryocyte marker). Each point in the scatter plots corresponds to a cell projected under SIMLR's 2-D embedding. The color of each point corresponds to the log10-expression value of the gene indicated in the subplot title. We observe that these genes are shared by a number of cells that are closely clustered in the 2-D embedding, which indicates the presence of the corresponding populations.

Nature Methods: doi:10.1038/nmeth.4207

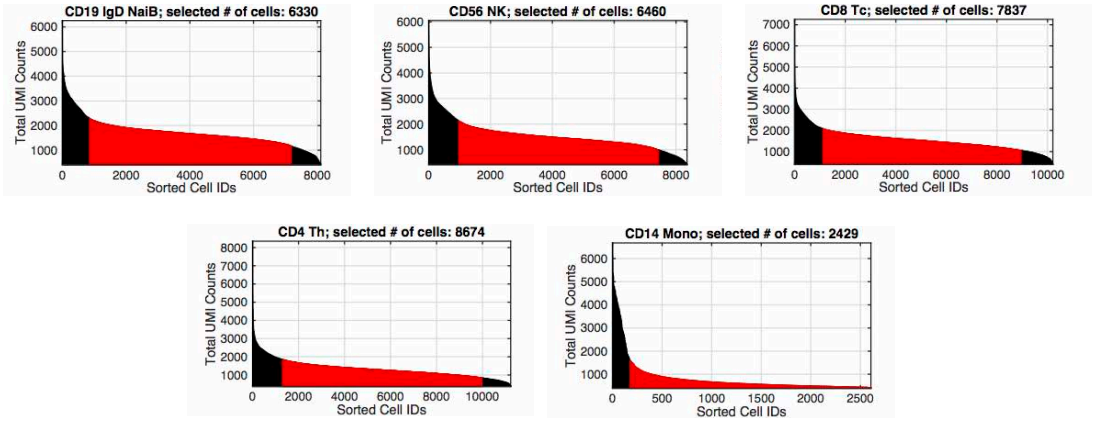

Figure 12: Selection of the cells from 5 purified populations from PBMC. The cells were sorted by total number of UMIs present and the cells colored red are the candidate cells used in the in silico experiment. 1000 cells total were drawn from these five populations at random at fixed proportions during each in silico trial.

Nature Methods: doi:10.1038/nmeth.4207

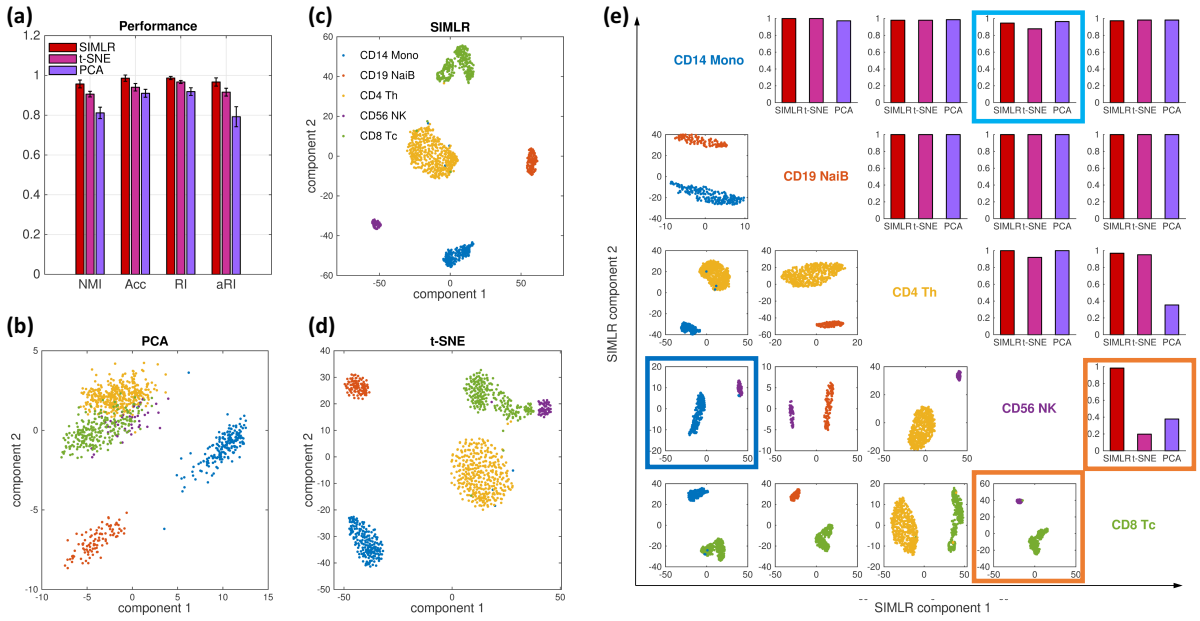

Figure 13: Comparison of SIMLR, t-SNE and PCA for visualizing and clustering the PBMC data set. (a) NMIs, Accuracy, Rand Index (RI) and adjusted Rand Index (aRI) between true cell identity and cluster labels for each method over 20 trials of randomly sampling 1000 cells from 5 immune cell populations at a fixed proportion. (b-d) 2D visualization with (b) PCA (c) SIMLR and (d) t-SNE for a single trial where scatter points are colored by true cell labels. (e) Pairwise comparison plots for a single trial where the row and column of a plot specifies the two cell populations (indicated on the diagonal) being compared. Plots in the upper triangle show the NMIs associated with SIMLR, t-SNE and PCA in clustering two different cell populations. Plots in the lower triangle show the 2-D projection of pairs of cell types using SIMLR, where points in different colors are of the cell type defined by the row and columns.

Nature Methods: doi:10.1038/nmeth.4207

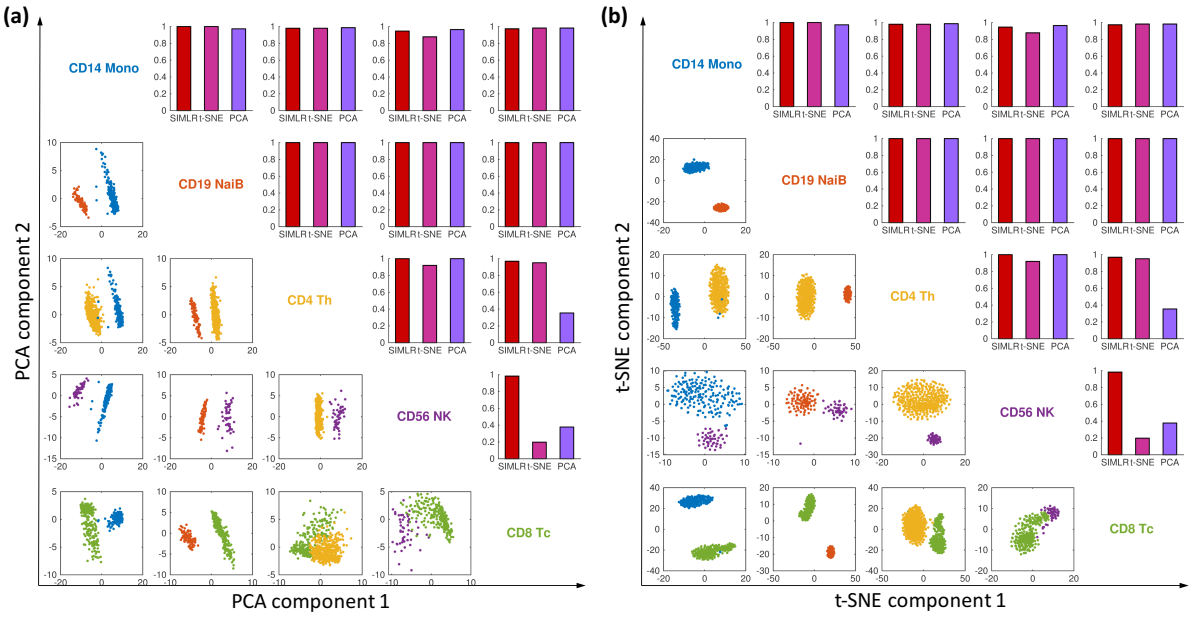

Figure 14: Pairwise comparison 5 immune cell subpopulations mixed in silico visualized with (a) PCA and (b)t-SNE in lower triangular plots.

Nature Methods: doi:10.1038/nmeth.4207

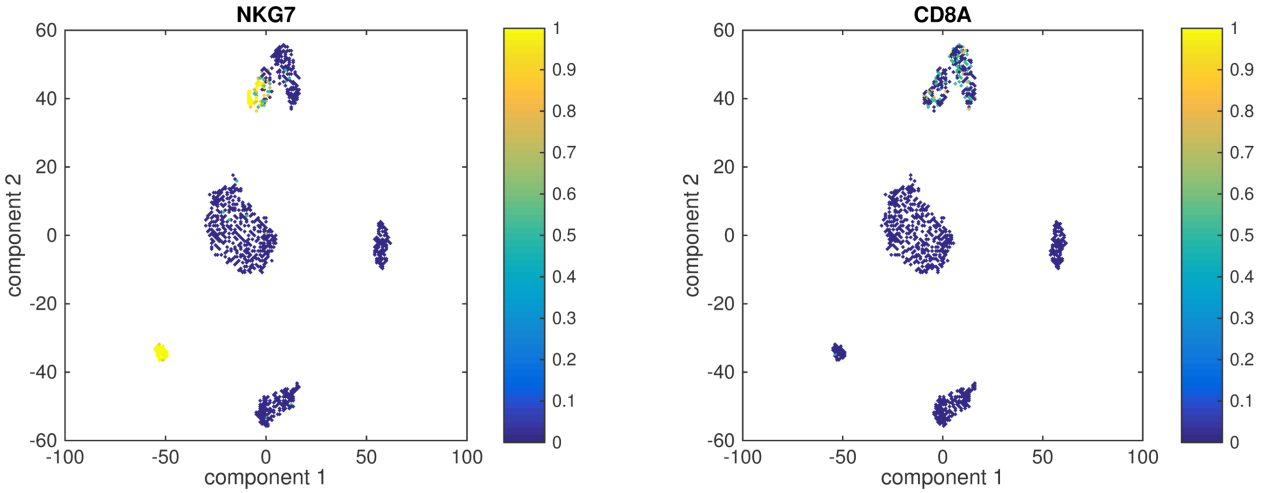

Figure 15: Gene marker visualization under SIMLR's 2-D embedding for 5 immune cell subpopulations mixed *in silico*. In top CD8+ T cell population, the left group is a population of activated CD8+ T cells, which exclusively express CD8A markers compared to other cell populations, and up-regulated NKG7 genes compared to the right group that expresses any NKG7 genes [13].

Nature Methods: doi:10.1038/nmeth.4207

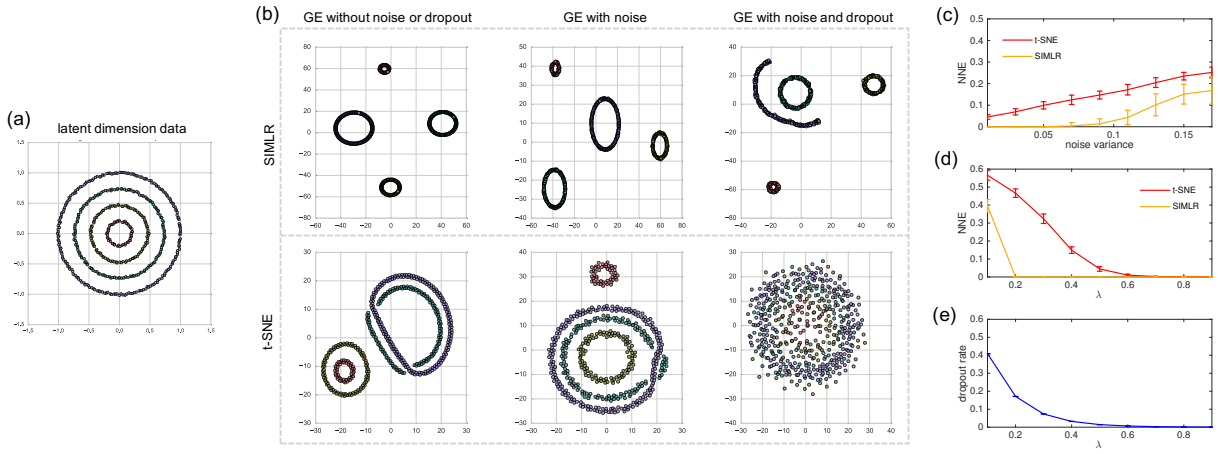

Figure 16: Dimension reduction of artificial gene expression generated from (a) 2-D latent data; each cluster is a concentric circle in the latent space. (b) The first row of plots show SIMLR's 2-D dimension reductions and the second row of plots show t-SNE's before adding noise or dropout (left plot), after adding noise (middle plot) and after adding noise and dropout (right plot). (c) NNE of t-SNE and SIMLR as noise is increased; lower scores denote better performance. (d) NNE of of t-SNE and SIMLR as zeros are added to the data (smaller values of λ indicate a higher dropout rate). (e) Fraction of entries zeroed out as a function of λ.

Nature Methods: doi:10.1038/nmeth.4207

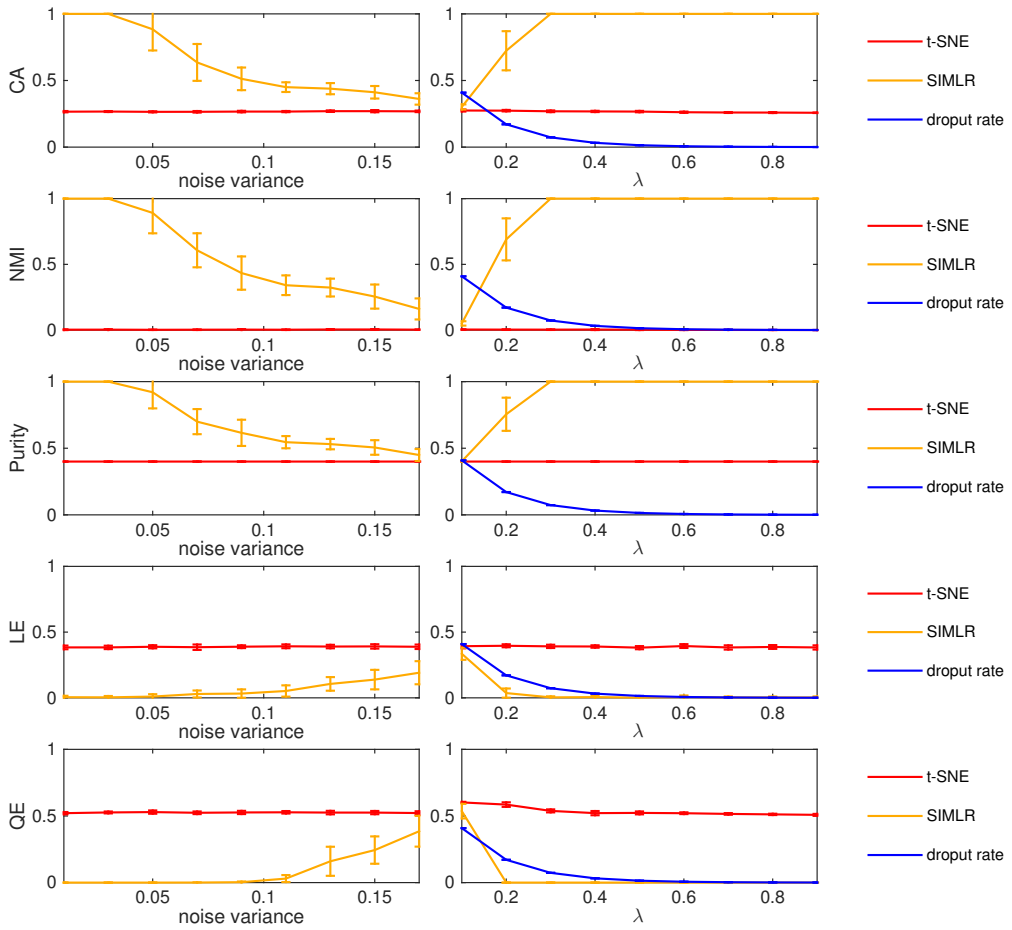

Figure 17: Sensitivity analysis of t-SNE and SIMLR on the simulated data set. Each row of plots corresponds to a clustering/ separability metric. The first column corresponds to the sensitivity of each method to the additive noise variance, and the second column corresponds to the sensitivity of each method to the decay coefficient which decreases as the dropout rate increases.

Nature Methods: doi:10.1038/nmeth.4207

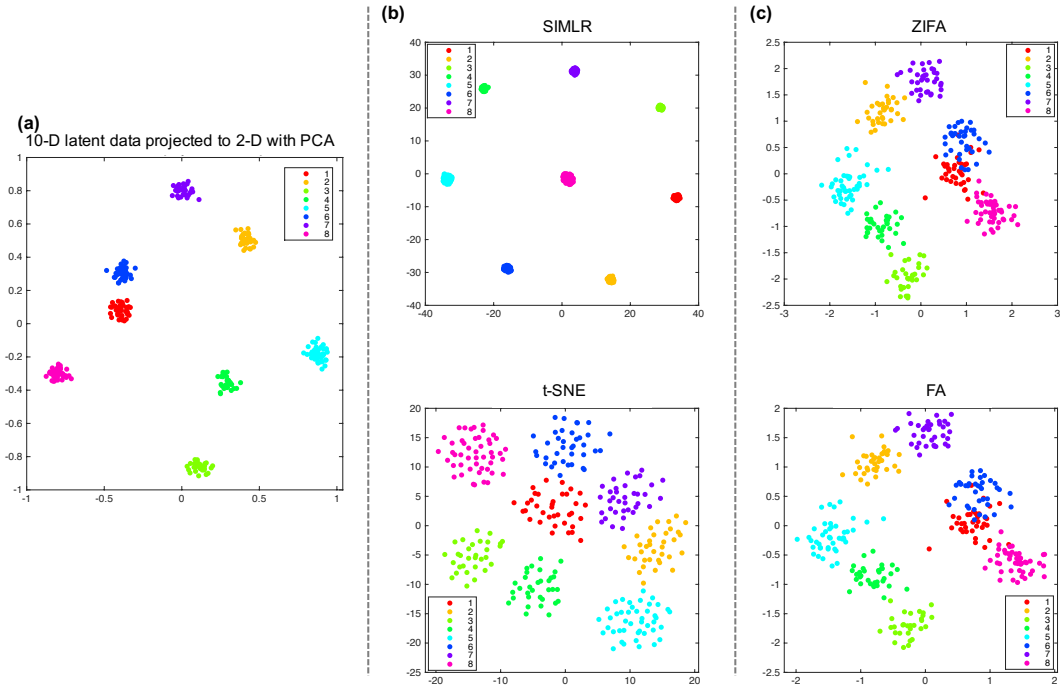

Figure 18: Simulated data set with 8 Gaussian clusters with low dropout rate (1%). (a) Latent data in the 10-D space visualized via PCA. (b) Dimension reduction of the simulated gene expression with similarity-based methods (c) Dimension reduction of the simulated gene expression with model-based methods. While no significant mixing of clusters is observed across all methods, SIMLR provides much more distinctive separation of the clusters. Without the true labels, the different classes would not be easily identifiable with t-SNE, ZIFA or FA. As expected, ZIFA and FA are similar when the dropout rate is low.

Nature Methods: doi:10.1038/nmeth.4207

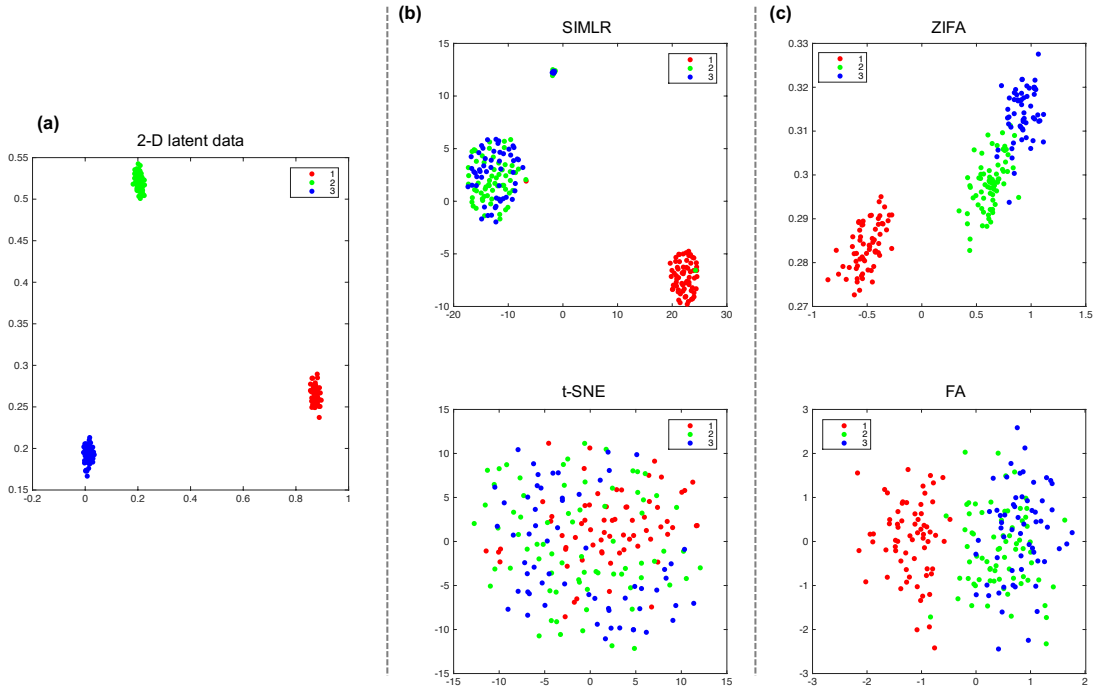

Figure 19: Simulated data set with 3 Gaussian clusters and high dropout rate (40%). (a) Latent data in the 2-D. (b) Dimension reduction of the simulated gene expression with similarity-based methods. (c) Dimension reduction of the simulated gene expression with model-based methods. Model-based methods (ZIFA and FA) retain the low dimensional Gaussian structure better than similarity-based methods (SIMLR and tSNE), and ZIFA which models dropouts provides more distinctive clusters than FA.

Nature Methods: doi:10.1038/nmeth.4207

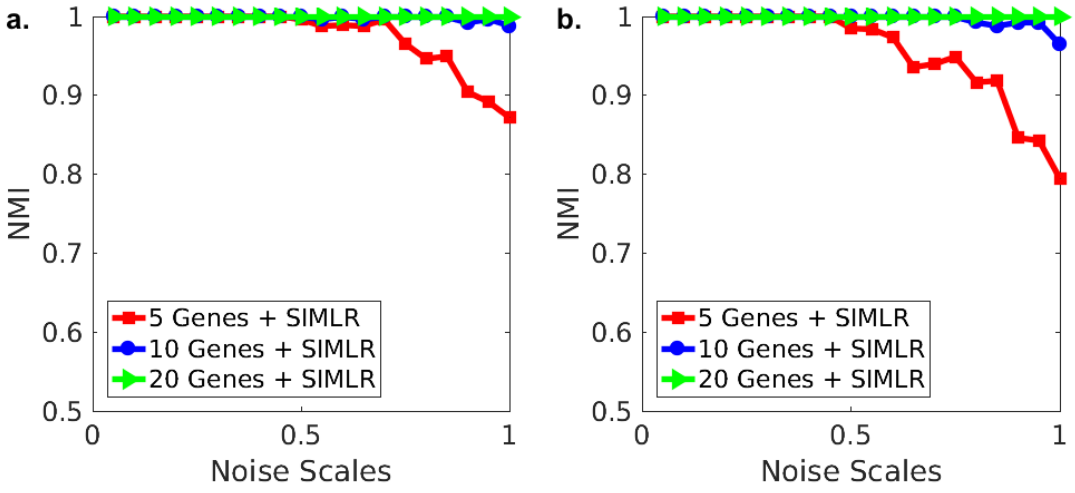

Figure 20: Results of simulations for low numbers of differentially expressed genes. (a) The Normalized Mutual Information (NMI) with different noise scales on small number of differentially expressed genes (5, 10, 20 genes respectively). (b) Results on same setting with the presence of background noise. Here noise scale measures the contents of corruptions on those few genes which are differentially expressed (Details of the simulation can be found in the appendix). We observe that SIMLR is able to properly address the issue raised by reviewer 2. As the number of differential genes increases, SIMLR can obtain better accuracy. Also, even with background noise in those similar genes, SIMLR can still obtain high accuracy.

Nature Methods: doi:10.1038/nmeth.4207

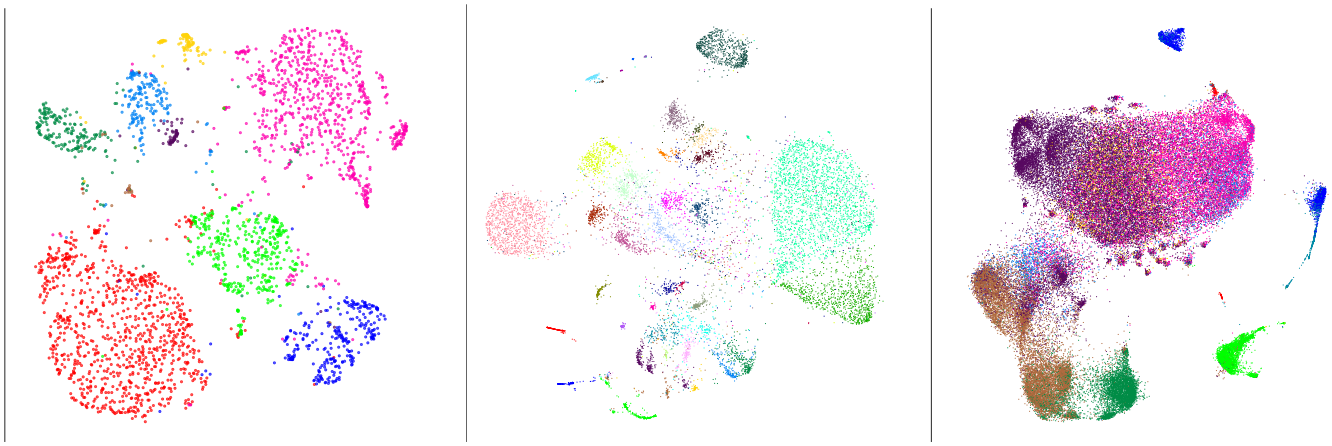

Figure 21: 2D visualization by SIMLR on three large-scale data sets: A) Zeisel et al ; B) Macosko et al. ; C) PBMC-68K. Each color represents originally assigned cell subpopulation.

Nature Methods: doi:10.1038/nmeth.4207

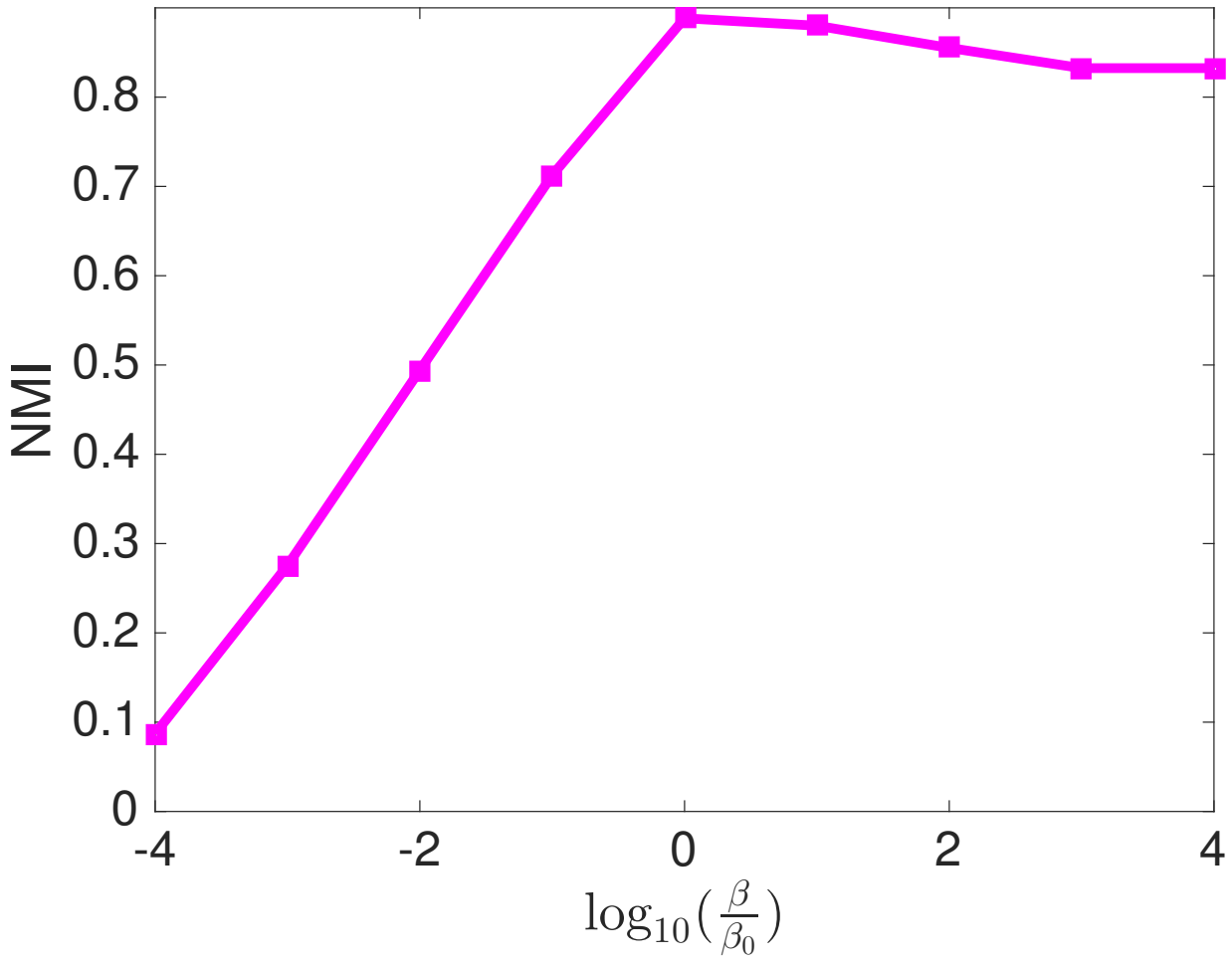

Figure 22: Effects of β on SIMLR. β serves as a regularization term to avoid the case where each cell only has very few non-zero similarities with others. β_0 is the default value which is estimated by the distance gap in a data-driven way (see Supp. Methods for details).

Nature Methods: doi:10.1038/nmeth.4207

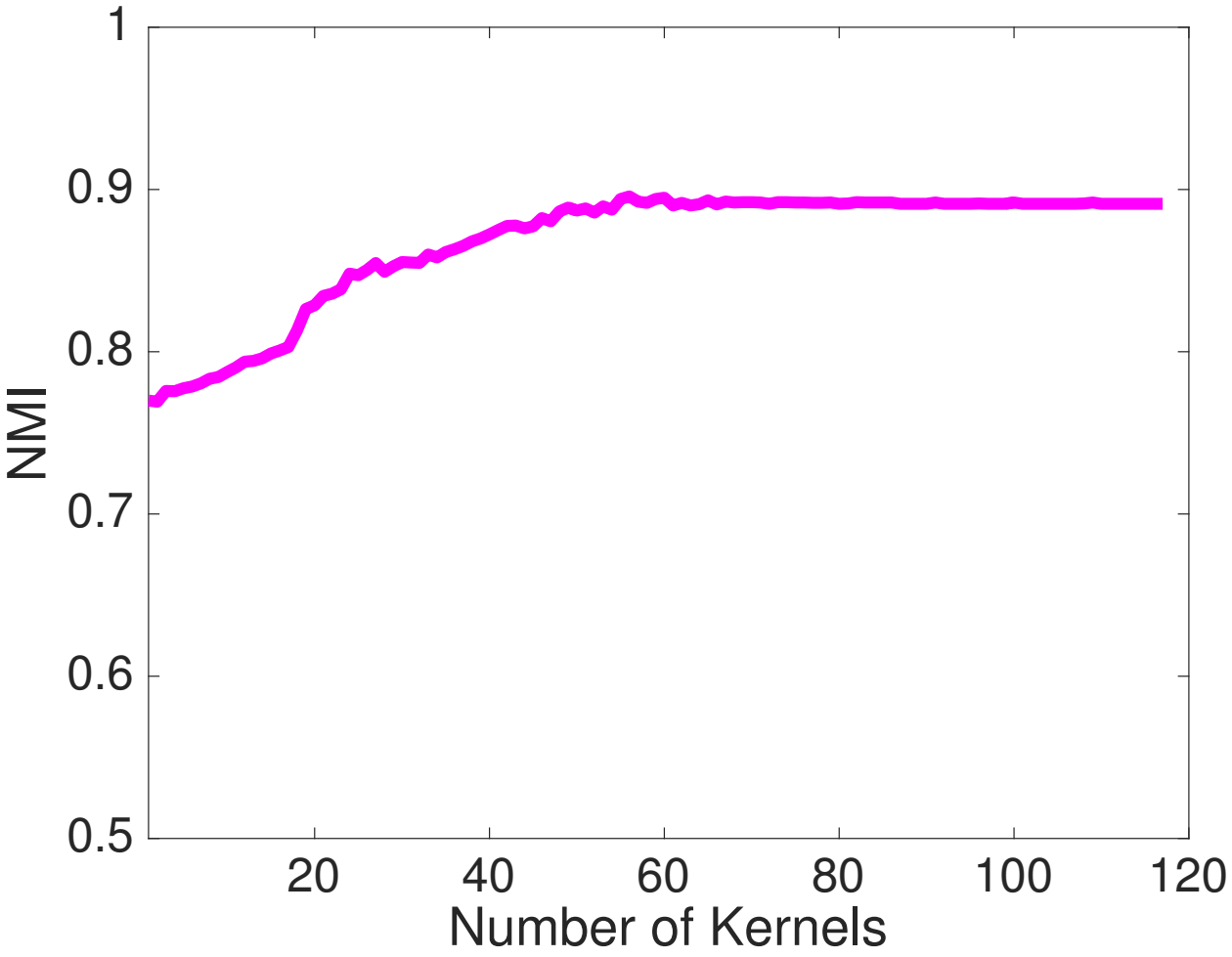

Figure 23: Effects of number of kernels on SIMLR. We use Buettner data set as an example. As number of kernels increases, the accuracy also increases. However, the accuracy saturates after a certain number of used kernels.

Nature Methods: doi:10.1038/nmeth.4207

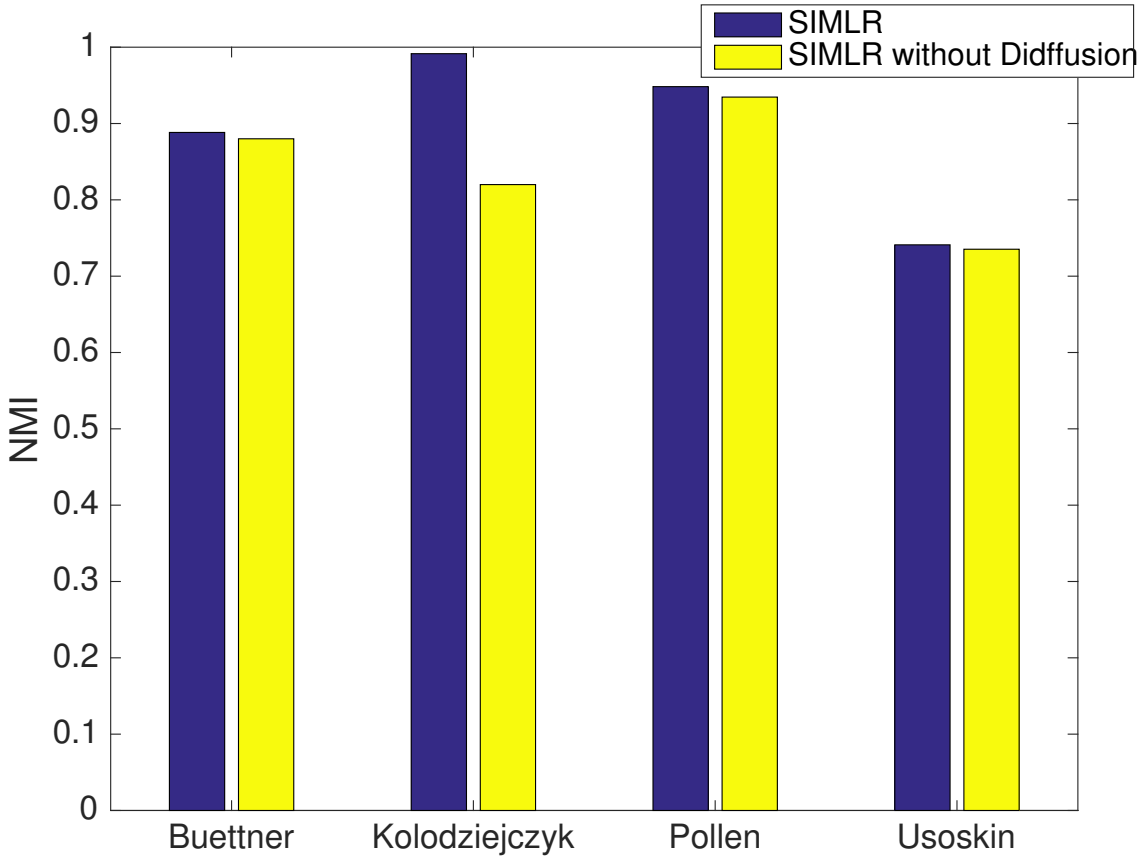

Figure 24: Effects of Similarity Diffusion for SIMLR on the four small scale single cell data sets.

Nature Methods: doi:10.1038/nmeth.4207

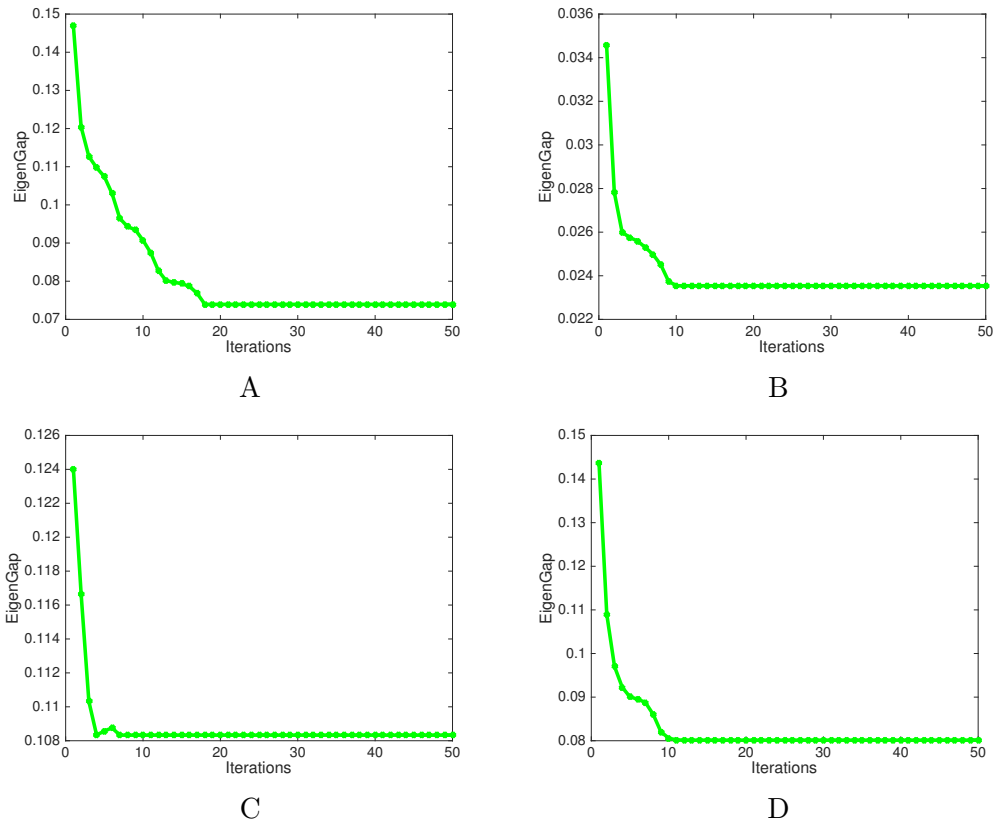

Figure 25: Convergence dynamics of the proposed method on the four real single-cell data sets: (A) Buettner [2], (B) Pollen [5], (C) Kolodziejczyk [6], and (D) Usoskin [7].

Nature Methods: doi:10.1038/nmeth.4207

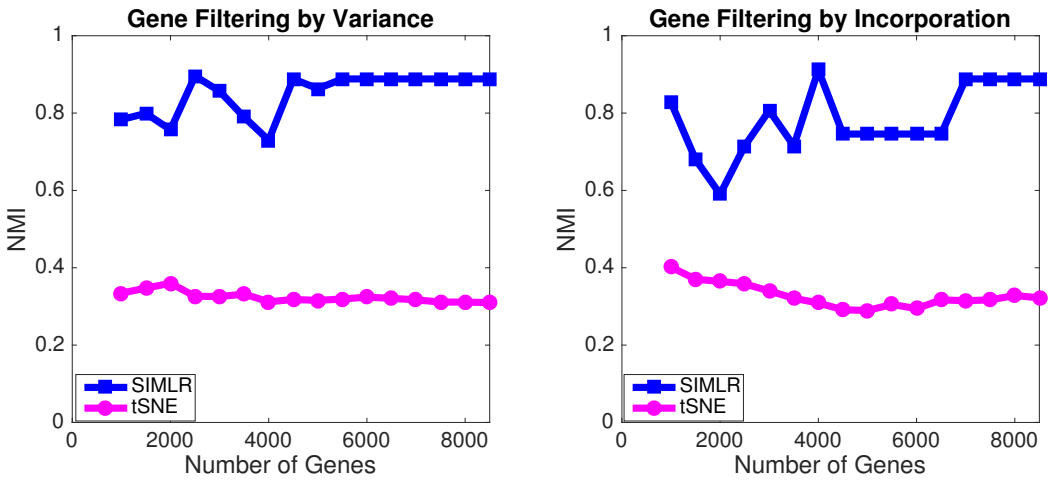

Figure 26: Effects of different gene filtering methods on SIMLR and tSNE on Buettner data set [2].

Nature Methods: doi:10.1038/nmeth.4207

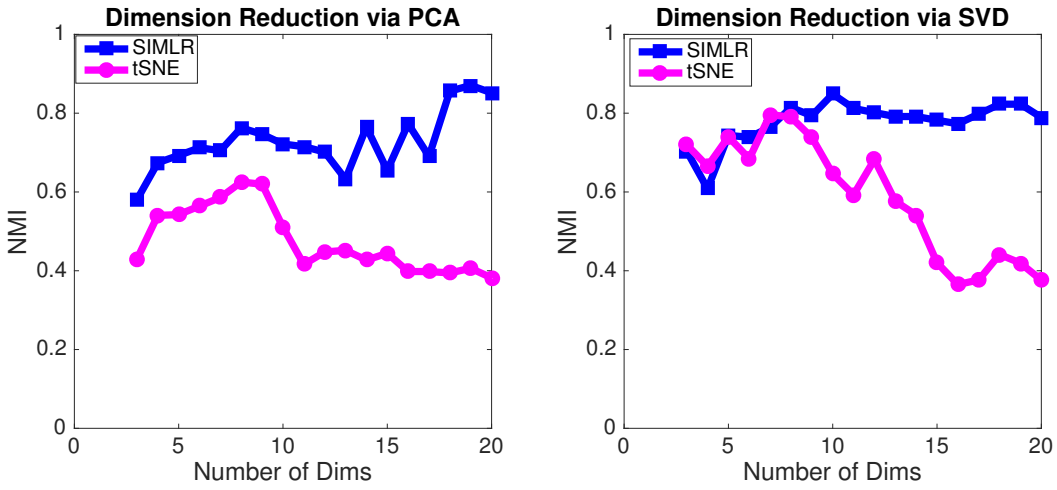

Figure 27: Effects of different dimension reduction methods on SIMLR and tSNE on Buettner data set [2].

Nature Methods: doi:10.1038/nmeth.4207

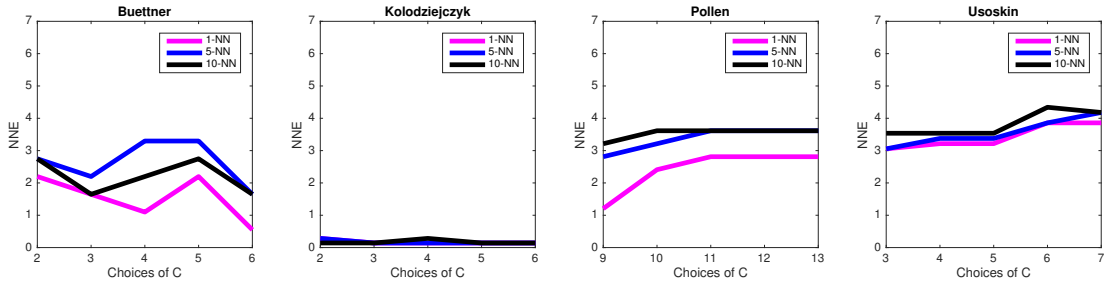

Figure 28: NNE values of different number of C with SIMLR on the four real single-cell data sets: Buettner [2], Kolodziejczyk [6], Pollen [5], and Usoskin [7].

Nature Methods: doi:10.1038/nmeth.4207

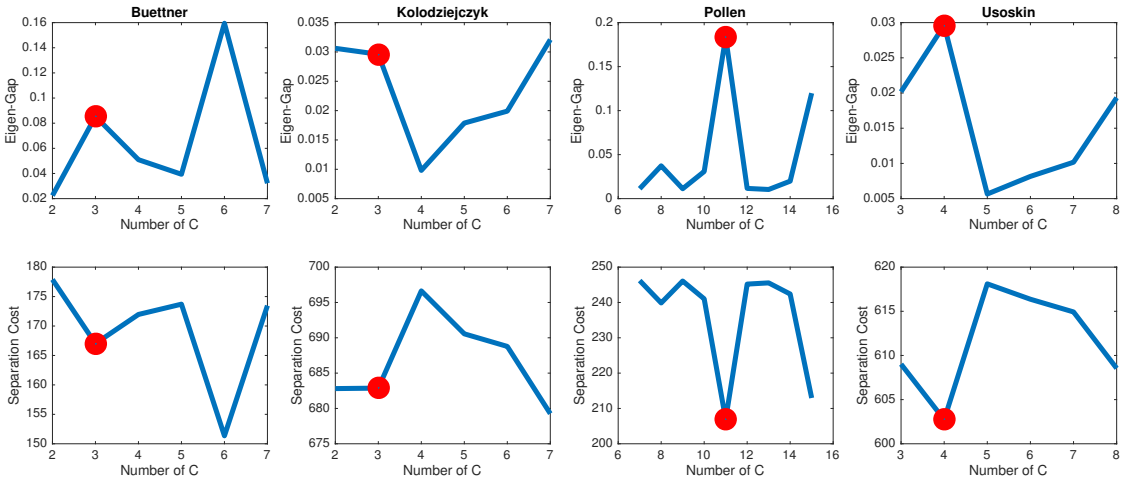

Figure 29: Two metrics of recommending the best choice of C with SIMLR on the four real single-cell data sets: Buettner [2], Kolodziejczyk [6], Pollen [5], and Usoskin [7]. The higher the eigengap is, the more likely the number is close to the optimal number of clusters. On the other hand, the smaller the separation cost is, the more likely the number corresponds to optimal number of clusters. The red dot indicates the optimal number of clusters for each data set. We can find that, these two metrics can provide a good estimate of the optimal number of clusters.

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Tables

Table 1: Summary of the characteristics of the four real single-cell data sets, with computational time for SIMLR.

Data Set	Number of Cells	Number of Genes	Number of Populations	Run Time (seconds)
Buettner	182	8989	3	2.369
Kolodziejczyk	704	10685	3	23.67
Pollen	249	14805	11	3.40
Usoskin	622	17772	4	18.54

Nature Methods: doi:10.1038/nmeth.4207

Table 2: NMI values for the four single-cell data sets to compare different dimension methods. Higher values indicate better performance.

Data Set	PCA	Laplacian	MDS	t-SNE	Sammon	PPCA	FA	ZIFA	SIMLR
Buettner	0.56	0.27	0.56	0.32	0.05	0.59	0.67	0.65	**0.89**
Kolodziejczyk	0.77	0.62	0.77	0.77	0.40	0.76	0.72	0.72	**0.99**
Pollen	0.83	0.80	0.83	0.93	0.75	0.83	0.82	0.79	**0.95**
Usoskin	0.39	0.48	0.39	0.69	0.41	0.40	0.36	0.41	**0.74**

Nature Methods: doi:10.1038/nmeth.4207

Table 3: NNE values for the four single-cell data sets to compare different dimension methods. Lower values indicate better performance.

Data Set	PCA	Laplacian	MDS	t-SNE	Sammon	PPCA	FA	ZIFA	SIMLR
Buettner	0.21	0.38	0.22	0.18	0.21	0.20	0.11	0.16	**0.05**
Kolodziejczyk	0.0016	0.0278	0.0016	0.0014	0.018	0.0016	0.0009	0.01	**0.0018**
Pollen	0.052	0.156	0.055	0.023	0.11	0.056	0.075	0.075	**0.02**
Usoskin	0.30	0.11	0.29	0.072	0.20	0.29	0.31	0.28	**0.063**

Nature Methods: doi:10.1038/nmeth.4207

Table 4: NMI values for the four single-cell data sets to compare different clustering algorithms. Higher values indicate better performance.

Data Set	NMF	GMM	DPMM	SIMLR	Euc.+AP	Corr.+AP	SIMLR+AP
Buettner	0.73	0.55	0.72	**0.89**	0.27	0.09	**0.94**
Kolodziejczyk	0.81	0.57	0.80	**0.99**	0.12	0.54	**0.80**
Pollen	0.93	0.91	0.84	0.95	0.80	0.88	**0.93**
Usoskin	0.62	0.31	0.65	0.74	0.31	0.51	**0.63**

Nature Methods: doi:10.1038/nmeth.4207

Table 5: Summary of six major cell types identified by unbiased analysis. We analyzed 2700 human PBMCs produced by 10x Genomics Gemcode Platform and identified the subtypes based on 5 most enriched genes specific to each cluster.

	Cluster Number	Gene Marker	Number of Cells	Proportion
Macrophages	1	LYZ+	688	25.48%
CD4+ T cells	2	LTB+	830	30.74%
B cells	3	CD74+	345	12.78%
Megakaryocytes	4	PF4+	12	0.44%
CD8+ T cells	5	NKG7+, CD3D+	470	17.40%
NK cells	6	NKG7+, CD3D-	355	13.14%

Nature Methods: doi:10.1038/nmeth.4207

Table 6: Summary and results on the three large-scale single-cell data. In Zeisel et al., we consider different levels of heterogeneity, which contain 9 and 48 sub-populations respectively. We report NMI values for four different methods.

Data Set	Number of Cells	Number of Clusters	PCA	t-SNE	SVD	SIMLR
Zeisel (level1)	3005	9	0.62	0.68	0.65	**0.77**
Zeisel (level2)	3005	48	0.58	0.57	0.52	**0.65**
Macosko	11040	39	0.66	0.59	0.69	**0.75**
PBMC(68K)	68560	10	0.51	0.43	0.48	**0.56**

Nature Methods: doi:10.1038/nmeth.4207

Table 7: Summary of statistics and time complexity for three large-scale data sets. We decompose SIMLR into three steps: 1) Learning cell-to-cell similarities; 2) Clustering cells into different number of populations; 3) Mapping cells into 2D or 3D space for visualization. The first step is the core of SIMLR algorithm and step 2 and step 3 are essentially applications of the similarities learned by SIMLR. Time is recorded in minutes and measured in MATLAB. Note that, for tSNE, we only report the time for visulization (i.e., 2-D embedding) because it is both memory and computationally intractable for tSNE to compute more than 2D embeddings in large scale datasets such as Macosko and PBMC68K.

data sets	Zeisel	Macosko	PBMC68K
Number of Cells	3005	11,040	68,560
Number of Clusters	9	39	10
Time for Similarity Learning	0.15	0.71	6.89
Time for Clustering	0.82	1.53	3.50
Time for Visualization	0.93	2.2	12.4
Time for tSNE Visualization	1.5	4.3	16.8

Nature Methods: doi:10.1038/nmeth.4207

Table 8: Effects of kernel weights regularization. We conduct experiments without any weight regularization ($\rho = 0$) and show the NMI values on four public data sets used in the main text. It indicates that adding weight regularization will enhance the performance of SIMLR.

	Buettner	Kolodziejczyk	Pollen	Usoskin
Single Kernel($\rho = 0$)	0.74	0.57	0.92	0.64
SIMLR	0.89	0.99	0.95	0.74

Nature Methods: doi:10.1038/nmeth.4207

Table 9: Summary of hyper-parameters used in SIMLR.

	C	B	k	τ	α
Explanation	num. of clusters	num. of embedding dim	num. of neighbors	smoothing parameter	moments
Recommended	data-dependent	close to C	$10 \sim 30$	$0.5 \sim 0.9$	$0.5 \sim 0.9$
Buettner	3	3	10	0.8	0.8
Pollen	11	11	10	0.8	0.8
Kolod	3	3	10	0.8	0.8
Usoskin	4	4	10	0.8	0.8
Zeisel	9	9	30	0.8	0.8
Macosko	39	39	30	0.8	0.8
PBMC	10	10	50	0.8	0.8

Nature Methods: doi:10.1038/nmeth.4207

Table 10: A summary of sparsity of single-cell RNA-seq data sets used in this paper.

Data Sets	Buettner	Kolodziejczyk	Pollen	Usoskin	Zeisel	Macosko	PBMC3k	PBMC68k
Sparsity	37.94%	27.87%	51.00%	78.10%	46.12%	68.85%	90.46%	91.92%

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 1: Gene Functional Analysis

Gene functional analysis typically involves an input gene set with jointly differentially up-regulated or down-regulated genes. Most single-cell differential gene analysis methods have adapted statistical and computational approaches developed for bulk RNA-seq data, where each class is known *a priori*. However, in heterogenous single cell populations, the class labels need to be identified through clustering, which heavily depends on the specific implementation and upstream processing prior to clustering, such as dimension reduction.

SIMLR's novel gene prioritization approach only relies on the output similarity S and does not depend on the downstream dimension reduction or clustering. The advantage of this step is that one can identify highly fluctuating genes that are consistent with the learned similarity in the multiple-kernel space. Genes with high variance or dispersion can be heavily influenced by both inter-class and intra-class variability. SIMLR reduces the noisy similarities that are present within each cell subpopulation and focuses on prioritizing genes that are strongly aligned with the expression in the learned similarity space. To show that the top prioritized genes can lead to meaningful functional gene candidates, we analyzed the Buettner data set [2] that was labeled by three ground truth cell cycle states (G1, S and G2M).

Because cell-cycle genes and proteins and the corresponding interactions have been widely studied in the literature, we were able to find remarkable interactions across the genes by considering the protein-protein interaction networks from the STRING database [1], especially when benchmarked against genes prioritized by variance and dispersion (Supplementary Figure 1). This contrast highlights the advantage of using a non-linear multi-kernel space as a surrogate to select genes that are most variable under the learned metric.

Further, the top prioritized genes can be used as candidate genes for the downstream differential gene analysis. To illustrate a simple example, we continued the analysis with the Buettner data set [2] using the ground truth labels for differential gene expression analysis using the top 500 genes prioritized by SIMLR. After assigning each gene to class-specific up-regulated and down-regulated sets (according to a two-class t-test statistic), we found 354 significantly up-regulated genes selected for the G2M phase and 145 significantly down-regulated genes selected for the G1 phase. (There was only one up-regulated gene specific to the S phase which we did not apply gene set analysis on.) Then we applied standard Gene Ontology (GO) analysis with g:profiler (using default settings [3]) on these two gene sets visualized by Enrichment Map (using default settings [4]) to find the significant ($p < 0.001$) biological processes that these gene sets define respectively. As expected, the GO analysis identified a large number of cell-cycle sub-processes as well as translation and process and metabolic signatures (Supplementary Figure 2).

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 2: Performance Metrics

Normalized Mutual Information

Throughout the paper, we commonly use the Normalized Mutual Information (NMI) [14] to evaluate the consistency between the obtained clustering and the true labels of the N cells. Given two clustering results U and V on a set of data points, NMI is defined as: $I(U,V)/\max\{H(U),H(V)\}$, where $I(U,V)$ is the mutual information between U and V, and $H(U)$ represents the entropy of the clustering U.

Specifically, assuming that U has P clusters, and V has Q clusters, the mutual information is computed as follows:

$$I(U,V)=\sum_{p=1}^{P}\sum_{q=1}^{Q}\frac{|U_p\cap V_q|}{N}\log\frac{N|U_p\cap V_q|}{|U_p|\times|V_q|}$$

where $|U_p|$ and $|V_q|$ denote the cardinality of the p-th cluster in U and the q-th cluster in V respectively. The entropy of each cluster assignment is calculated by

$$H(U)=-\sum_{p=1}^{P}\frac{|U_p|}{N}\log\frac{|U_p|}{N},$$

and

$$H(V)=-\sum_{q=1}^{Q}\frac{|V_q|}{N}\log\frac{|V_q|}{N}.$$

Further details on NMI can be found in [15]. NMI takes on values between 0 and 1, measuring the concordance of two clustering results. In the simulation, we calculated the obtained clustering with respect to the true labels. Therefore, a higher NMI refers to higher concordance with ground truth, i.e. a more accurate label assignment of each cell.

Clustering Accuracy

Given a cell x_i, let u_i and v_i be the label output from an algorithm and the true label, respectively. The Clustering Accuracy (CA) is defined as:

$$CA=\frac{\sum_{n=1}^{N}\delta(v_i,map(u_i))}{N}$$

where N is the number of cells; $\delta(x,y)$ is the delta function that equals one if $x=y$ and equals zero otherwise, and $map(u_i)$ is the mapping function that maps each assigned label u_i to the equivalent true label. The best mapping can be found by implementing the Kuhn-Munkres algorithm [16].

Purity

Compared to more complicated mapping methods such as Clustering Accuracy, Purity is another simple and transparent metric used to measure clustering performance. For purity, each identified cluster is assigned to the label which is most frequent in the cluster, and then the accuracy of this label assignment is computed by counting the number of correctly assigned cells and dividing by

Nature Methods: doi:10.1038/nmeth.4207

the number N. This mapping is not one-to-one and may be biased to the class which has the largest size. Nonetheless, it provides us a simple metric:

$$purity(U, V) = \frac{\sum_p \max_q |U_p \cap V_q|}{N}.$$

Similar to NMI, Purity is also a value between 0 and 1 and a higher purity indicates a better consistency between the assigned labels and the true labels.

Adjusted Rand Index

The Adjusted Rand Index (ARI) is another widely-used metric for measuring the concordance between two clustering results. Given two clustering U and V, we calculate the following four quantities:

a: number of objects in a pair are placed in the same group in U and in the same group in V;
b: number of objects in a pair are placed in the same group in U and in different groups in V;
c: number of objects in a pair are placed in the same group in V and in different groups in U;
d: number of objects in a pair are placed in different groups in U and in different groups in V .

The (normal) Rand Index (RI) is simply $\frac{a+d}{a+b+c+d}$. It basically weights those objects that were classified together and apart in both U and V . There are some known problems with this simple version of RI such as the fact that the Rand statistic approaches its upper limit of unity as the number of clusters increases. With the intention to overcome these limitations, ARI has been proposed in [17] in the form of

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]}.$$

Nearest Neighbor Error

The three metrics proposed above are used in unsupervised settings where the true labels are not used in obtaining the cluster labels. We also consider a number of performance metrics in a supervised setting: We use 10-fold cross validation and adopt simple classifiers. On each trail of cross-validation, we use 9 folders as training data, and the remaining one folder as validation data. The average of the ten validation errors is recorded to show how the obtained latent features can facilitate the classification task. The simplest classifier is nearest neighbor classifier, and the corresponding error is called as Nearest Neighbor Error (NNE). Given a validation data set, X_V and a training data set X_T, we first calculate the Euclidean distance between X_V and X_T, then for a cell x_v in $X_V s$, the assigned class is the label of the cell in X_T which has the smallest distance to x_v. The final NNE is defined as the average classification error on the ten folds. Since there is randomness in the 10-fold cross validation setting, we perform the cross validation independently for 20 times. That is, 20 different cross validation data splits are used and on each split, we report the average validation error from the 10 folders. Therefore, the final NNE error is the average of $20 * 10$ validation errors. Note that NNE error is a direct measure of the goodness of the distance measure from the latent features. A good latent feature should facilitate such distance-based classifiers.

Nature Methods: doi:10.1038/nmeth.4207

Linear Error

In addition to a simple distance-based classifier, we also consider a linear classifier. We adopt a linear support vector machine (SVM) classifier which aims to fit a linear boundary function:

$$y = w^T x + b$$

where w is a linear function and b is an offset. We use a fast implementation of linear SVM from [18]. It uses stochastic gradient descent to learn the linear SVM. Due to the existence of randomness in the SVM solver, we also average such computation for 20 times and average the final error.

Quadratic Discriminative Error

Beyond a linear classifier, we also consider a non-linear classifier: quadratic discriminative analysis (QDA). QDA models a multi-variant gaussian likelihood function in the following form:

$$p(X|y=k) = \frac{1}{(2\pi)^d |\Sigma_k|^{1/2}} \exp(-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1}(X-\mu_k))$$

Notice that QDA does not assume the covariance matrices Σ_k to be diagonal, resulting a quadratic decision boundary.

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 3: Dimension Reduction Comparisons

To analyze well how SIMLR performs dimension reduction, we compare SIMLR's dimension reduction application to other dimension reduction methods by comparing the latent space projection produced by SIMLR to the latent space projections produced by other methods. Following a supervised approach used by [19] to compare different dimension reduction methods, we classify each cell based on the true labels of its nearest neighbors in the dimension-reduced space, and assess the accuracy of this classification using cross validation; we refer to this metric as nearest neighbor error (NNE) (Supplementary Note 2). This test assesses how accurately new cells can be classified using cells whose labels are already known. In addition to using the NNE, we also compare different dimension reduction results following an unsupervised approach: we use K-means to cluster cells in these different latent spaces and assess which latent-space clustering has the greatest concordance with the true labels. When comparing different dimension reduction methods combined with K-means, we assume the number of clusters C is known a priori. The performance metric we use to compare the results to the true cell identities is the normalized mutual information (NMI, Supplementary Note 2), a standard measure of clustering concordance. We select these two types of performance metrics for dimension reduction specifically because they measure different aspects of the data in the low dimensional latent space: NNE directly reflects how closely cells from the same population are surrounded by each other, whereas NMI provides a global view of how well cells from different populations are separated. We also tested four other similar performance metrics in Supplementary Note 2, shown in Supplementary Figures 3-6, to ensure that SIMLR performed well under different standards. We performed extensive comparisons of SIMLR with 8 other dimension reduction methods on the four data sets to test its utility for dimension reduction. The 8 methods included standard linear methods including PCA, Factor Aanalysis (FA), and probabilistic PCA (PPCA); nonlinear methods including t-SNE, Laplacian eigenmaps, multidimensional scaling (MDS), and Sammon mapping [20]; and model-based methods specifically designed for single-cell data like zero-inflated factor analysis (ZIFA) [19]. The methods we tested included all methods used in analyses of the original data sets. In addition, the Pollen and the Usoskin data sets were the only ones with validated labels used in [19] to assess ZIFA, which was specifically designed for single-cell data. The NMI and NNE values for the 9 methods are summarized in Supplementary Table 6 and 3, respectively. SIMLR consistently outperforms the existing alternatives on the four data sets, and most of the differences in NMI and NNE between SIMLR and the second best method are remarkably large. To test the robustness of SIMLR, we conducted three additional sensitivity experiments for each data set. We used varying numbers $(3-20)$ of latent dimensions B to evaluate the performance of SIMLR and other methods. This evaluation is critical because typically the true number of clusters in the data is unknown. (As mentioned before, ideally B should be equal to the true number of clusters for SIMLR.) We dropped varying fractions $(5\% - 70\%)$ of the gene measurements in the input gene expression matrix to analyze how each method performs when random levels of dropout are present across the data, which is relevant to the high dropout rate in single-cell data. We added independent zero-mean Gaussian noise with varying variances to the gene expression matrix. The number of dimensions C used in this experiment is set to be the true number of clusters. The number of dimensions C used in this experiment is set to be the true number of clusters. In order to preserve the dropout characteristics, we ensured that the added noise was set to zero at a frequency equal to the dropout rate in each data set. Formally, we added

Nature Methods: doi:10.1038/nmeth.4207

a random noise vector y_i to x_i by the following process:

$$z_i \sim \text{Normal}(0, \sigma^2 I_M), h_{ij} \sim \text{Bernoulli}(p_0), y_{ij} = \mathbb{I}_{\{h_{ij}=0\}} z_{ij}$$

where p_0 is the dropout rate (proportion of zeros) in the original expression matrix X. We set the values of an entry in the new expression matrix to zero if its value after adding noise dropped to zero or below. The NMI and NNE values (Supplementary Table 6 and 3) on the Buettner data set show remarkably better performance by SIMLR as compared to other methods. In addition, we observe that SIMLR is not sensitive to the number of latent dimensions B even though the most suitable choice of B should ideally be the number of clusters C if it is known. ZIFA and FA achieve high NMI values at certain values of B but are less stable and can perform much worse than SIMLR otherwise (as is shown in the first column of Figure 3). Moreover, as the fraction of genes increases, the NMI increases and the NNE decreases more noticeably for SIMLR. Finally, while the performance of SIMLR decreases with the noise variance, it still outperforms other methods. Based on other metrics, including Clustering Accuracy and Purity, on all four data sets (shown in Supplementary Figures $3 - 6$), we observe that SIMLR also outperforms other methods on the three sensitivity experiments above and no other method dominates in specific regimes.

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 4: Clustering Comparisons

Comparison with different clustering algorithms

To assess SIMLR's clustering application, we used the following two-stage procedure: (1) reduce the data to a C-dimensional latent space using SIMLR with input parameter C, and (2) apply K-means clustering with input number of clusters equal to C to compute the cluster assignment of each cell. However, because SIMLR is a similarity framework, it can be flexibly adopted in other clustering frameworks that directly take similarities as inputs. So we also assess SIMLR combined with Affinity Propagation (AP) [21] (rather than k-means). We found that K-means [22] clustering with SIMLR consistently outperforms the existing alternatives: Non-Negative Matrix Factorization (NMF) [23], Gaussian Mixture Models (GMM) [20], Dirichlet Process Mixture Models (DPMM) [24], on the four data sets (Supplementary Table 4). These three clustering methods are model-based algorithms that have been used in recent single-cell studies [25]. Further, we conducted extra experiments with AP which takes similarities as inputs to demonstrate that the similarities learned by SIMLR (used for SIMLR+AP) can significantly outperform the other simpler similarities, such as Euclidean similarity (used for Euc.+AP) and Pearson correlation (used for Corr.+AP). This superior clustering performance is expected from plotting of the similarity matrices (Figure 2 in main text). Clustering methods typically require the number of clusters as an input parameter in order to reveal meaningful grouping in the data. Thus, when applied to clustering, the input parameter C to SIMLR can be conveniently assigned to be the number of clusters used as an input for the downstream clustering algorithm. However, if number of clusters is not known a priori, SIMLR provides two heuristics based on the structures in the similarities to estimate an optimal choice of C (Supplementary Note 9). Effectiveness of these two heuristics is demonstrated on the four single-cell datasets (Supplementary Figures 28 and 29).

Comparison with Matrix Completion Methods

The problem of Matrix Completion (MC) is to recover a matrix A from only a small sample of its entries. Matrix completion has been widely used in recommendation systems. It takes a subset of observed data as input and aims to impute unknown entries based on low-rank assumptions [26]. Though matrix completion has not been used in single-cell analysis, to our best knowledge, we still consider it a potential way to overcome high dropouts in scRNA-seq Data. We adopt three well-known matrix completion methods: Inexact Augmented Lagrange Multiplier (IALM) [26], Gradient-based Augmented Lagrange Multiplier (GALM) [27] and Singular Value Thresholding (SVT) [28] . We take the implementations directly from the authors.

Matrix completion methods result in a dense matrix whose unknown entries are imputed. We use the resulted dense matrix and compare four dimension reduction methods: SIMLR, tSNE, PCA and NMF. We report the results on the four public data sets in Supplementary Figure 8. It can be observed that, matrix completion decreases the performance of most of the dimension reduction methods, compared with original sparse gene expressions. There are three potential explanations: 1) matrix completion assumes low rank on the gene expression which may not be applicable in single-cell data; 2) matrix completion minimizes the trace norm of gene expression, and this objective can be easily distorted since the original gene expression matrix is highly sparse; 3) sparsity in gene expression may also contain useful information about cell heterogeneity while matrix completion fails to capture such information by wrong or random imputations.

Nature Methods: doi:10.1038/nmeth.4207

Effect of Binarization and SVD on single-cell RNA-seq

Here we consider two simple transformation forms of gene counts before applying any single-cell analysis algorithm.

(T1) Binarizing the input gene expression data.

(T2) Binarizing followed by SVD.

T1 is simply setting nonzero gene counts to 1. This would simplify the input and hopefully remove some outliers in gene counts. T2 is an extra step to recover the main structural information by applying SVD on simplified binarized data. We test the effects of these two operations on four public single-cell data sets. Results are reported in Supplementary Figure 9. It is observed that, SIMLR, under these two transformations, still outperforms the rest alternatives. Also, binarization of single-cell RNA-seq usually would decrease the performance, especially for tSNE. However, T2 seems to be more helpful than T1 in that it extracted the main cluster information. In general, T1 and T2 would not improve the performance significantly and therefore we would not recommend such transformation before any analysis.

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 5: Rare populations in PBMC data

To demonstrate how SIMLR can be used for *de novo* cell type discovery, we analyzed 2700 human PBMCs generated by the 10x Gemcode Platform [25]. The data was processed through the 10x Cell Ranger pipeline. We used the the Cell Ranger R kit [25] to normalize and log-transform the UMI counts per cell before applying a filter to select the genes that had variance greater than 0.001 across all cells.

Next, we ran SIMLR on the expression matrix with $C = 5, 6, 7$, which is the number of major cell types we expected. Because SIMLR's default implementation involves k-means clustering as the downstream clustering, we set the number of clusters to be C. After clustering, we took the mean expression for each cluster and compared the expression of a specific cluster against the mean expression of the other cells. The genes were sorted by the enrichment fold change for each cluster.

Using $C = 6$ and the simple fold-change test above, we were able to identify each cell type based on the top 5 gene markers in Supplementary Table 5: macrophages, CD4+ T cells, metakaryocytes, NK cells, CD8+ T cells, and B cells (Supplementary Figure 10a). The top genes specific to each cluster (Supplementary Figure 10b) were very distinct to each population. These cell populations were also validated by plotting specific gene markers of interest across all the cells (Supplementary Figure 11). The unbiased analysis without significant fine tuning suggests that SIMLR's framework is robust and sensitive to populations that can occur at lower frequencies (e.g., metakaryocytes that are less than 1% abundant in the a PBMC sample).

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 6: Simulation Experiments

In silico PBMC Simulation

The PBMC data were from 5 bead enriched populations generated by 10x Genomics [25]: naïve B (CD19+ and IgD+), CD56+ natural killer (NK) cells, CD8+ cytotoxic T cells, CD4+ T cells and CD14+ monocytes. The purity of the populations was validated by FACS, and clustering analysis of the scRNA-seq transcriptome profile. We computationally sampled a total of 1000 cells randomly from the individual purified populations at the proportion of 5%, 10%, 25%, 40%, and 20% respectively for each *in silico* trial after selecting cells by total UMI counts (Supplementary Figure 12).

SIMLR provided an unbiased classification of these subpopulations that was highly consistent (with NMI over 0.95 on average). In addition, we used t-SNE and PCA with K-means clustering as baselines, and they also produced good agreement with the true labels (Supplementary Figure 13a). The overall improvement of SIMLR over t-SNE and PCA was noticeable but not highly significant because many cell types (such as monocytes, naïve B cells) are easily distinguishable from other cell types.

To provide intuition for how SIMLR differs from t-SNE and PCA, we illustrate one of the trials (Supplementary Figure 13b-c, colored by ground-truth cell types) where the 5 different cell types are better separated using SIMLR. PCA produced the most ambiguous visualization across NK cells, CD8+ cytotoxic T cells and CD4+ T cells, whereas t-SNE did not completely separate NK cells from CD8+ cytotoxic T cells. We performed pairwise clustering to elucidate the cases where SIMLR and other methods differ (Supplementary Figure 13d and Supplementary Figure 14). While most pairs were easy to separate by all methods, SIMLR's overall improvement over t-SNE and PCA resulted from its ability to separate CD8+ T cells and NK cells (highlighted in the orange boxes in Figure 5e). CD8+ T cells and NK cells are difficult to separate because the two cell types can share several common gene markers and certain T cells share properties of both NK and CD8+ cells [29], [30]. In the case where SIMLR did not outperform PCA (highlighted in the blue boxes in Figure 5d), we found that SIMLR mistakenly grouped a very small number of monocytes with NK cells but still correctly separated the vast majority, leading to a negligible difference from PCA.

With input parameter $C = 5$ for this *in silico* experiment, we were still able to see additional sub-structures underlying the major populations. For instance, SIMLR revealed separation among the CD8+ T cells (Supplementary Figure 13c) into two main groups. We found the left group to be activated CD8+ T cells which show up-regulation of NKG7 genes compared to the right group that expresses any NKG7 genes [13] (Supplementary Figure 15) . Therefore, this biological interpretation indicates that SIMLR is able to identify meaningful sub-populations.

Simulations with generative models

In order to supplement the analysis of real single-cell data, we generated simulated data to evaluate the performance of dimension reduction applications followed by SIMLR where we have full ground-truth information. Similar analysis has been used by [19] to distinguish the performance of different dimension reduction methods.

There are four steps in the simulation, and each step can be qualitatively and quantitively analyzed using any dimension reduction methods that one wishes to compare.

Nature Methods: doi:10.1038/nmeth.4207

Step 1: Generate latent data Z^{true} in the B-dimensional latent space. In the experiments we conducted, we used two types of Z^{true}: (i) concentric circles and (ii) Gaussian mixture in 2-D latent space. (The implementation details are provided in the next subsection.) Step 2: Project the latent data to a large dimension space using a random projection matrix $P \in R^{B \times M}$, with independent uniformly distributed entries, i.e., $P_{i,j} \sim Uniform(0,1)$. The number genes, M, is usually a large number. The resulting gene expression data $X^{true} \in R^{N \times M}$ is given by

$$X^{true} = Z^{true} P.$$

Step 3: Add independent Gaussian noise to each entry of X^{true} to simulate a noisy gene expression matrix X^{noise}. Thus, each entry is updated according to

$$X_{i,j}^{noise} = X_{i,j}^{true} + \mathcal{N}(\mu, \alpha)$$

where $\mathcal{N}(\mu, \alpha)$ represents random gaussian noise with mean μ and variance α.

Step 4: Introduce dropout events according to the double-exponential model for a given λ to produce the final gene expression X.

$$X_{i,j} = X_{i,j}^{noise} \delta\{q_{i,j} \geq \exp -\lambda (X_{i,j}^{noise})^2\}$$

where $q_{i,j} \sim Uniform(0,1)$and δ denotes the Kronecker delta, which is one if its argument is true and zero otherwise.

The latent data Z^{true} was generated in different ways for two different experiments to represent structures in the data where most common dimension reduction methods have challenges handling. Our analysis also included the same latent space model proposed by [19].

- *Concentric circles in the latent dimension:* In the 2-D latent space (i.e., using $B = 2$), $N = 1000$ points were generated to create $C = 4$ concentric circles, each of which is considered to be one cluster associated with a distinct color. The number of points in each class was set to be proportional to the radius of the corresponding circle. To each point, we added additional independent noise with zero mean and variance 0.005 in each dimension in the 2-D latent space. The number of genes was set as 5000. For the additive noise, we had $\mu = 3$ and $\alpha = 0.1$. A decay rate of $\lambda = 0.2$ for the exponential model was used.

- *Gaussian mixtures in the latent dimension:* For one example with 3 clusters (Supplementary Figure 19), we set $C = 3$, $B = 2$. Each Gaussian in the latent dimension had a mean uniformly distributed independently between 0 and 1 in each dimension, the variance of each cluster was set to 0.0001 for each dimension. The noise and dropout parameters used were $\mu = 3$, $\alpha = 0.1$, and $\lambda = 0.1$.
  For another example with 8 clusters (Supplementary Figure 18), we set $C = 8$, $B = 10$. Each Gaussian in the latent dimension had a mean uniformly distributed independently between 0 and 1 in each dimension, the variance of each cluster was set to 0.001 for each dimension. The noise and dropout parameters used were $\mu = 3$, $\alpha = 0.8$, and $\lambda = 0.8$.

We first focused on comparisons to t-SNE, an algorithm designed to capture nonlinear manifolds, which SIMLR directly adapts by incorporating the learned similarity matrix into the t-SNE

Nature Methods: doi:10.1038/nmeth.4207

framework. Artificial clusters of cells (concentric circles) were created in a low-dimensional latent space (Supplementary Figure 16a).SIMLR separates the clusters more clearly than t-SNE (Supplementary Figure 16b), as measured by NNE (results were similar with other metrics;Supplementary Figure 17)at each stage of the data generation process: prior to adding noise or zeros, after adding noise, and after adding zeros.When morenoise(Figure fig:circlesc)or dropouts(Figure fig:circlesd)are added to the data, the cluster information in t-SNE becomes obscured, whereas the impact on SIMLR is less significant as the NNE is lower and the clusters are still separated very well.

While in this example we generate clusters as concentric circles in the latent space to illustrate how SIMLR can separate even clusters with complex structures, we also evaluated SIMLR's performance on Gaussian clusters (Supplementary Figures 18 and 19), comparing to factor analysis and ZIFA [19] as well as to t-SNE. We found that SIMLR outperformed the other methods if the dropout rate was low, whereas ZIFA outperformed the other methods if the dropout rate was high; the latter result is unsurprising, because ZIFA is explicitly designed to accommodate the dropout problem and we generated the data using ZIFA's dropout model.

Simulations with extremely low signal-to-noise ratio

Note that Gaussian kernels have already been widely used and proven to be effective in single-cell RNA-seq analysis ([31], [32]), because Gaussian kernels, by exponentiating pair-wise distances, are able to capture and amplify the weak signals hidden in the high-dimensional space. In the presence of background noise, the genes that are not differentially expressed will contribute the same amount on average to the kernels for cell-to-cell comparisons. Thus, as long as the sc-RNAseq data contains the signals of the few differentially expressed genes, we can still expect to observe the discrepancy among the kernels. SIMLR, adopting multiple Gaussian kernels, learns a suitable combination of Gaussian kernels where each kernel further amplifies the signals discriminating between different cell types. Thus, overall the difference between among the few genes can be captured.

In addition to the Gaussian kernels, SIMLR enforces a low-rank constraint on our learned cell-to-cell similarity. The low-rank constraint can also help amplify the differences between cell types if there are indeed two populations that differ at only a few genes. Further, our network enhancement module will enlarge the cell population-level variability since it imputes the missing links between cells by diffusing local similarities. These two additional novel components allow SIMLR to successfully handle cases even more complicated cases.

To empirically demonstrate the arguments above, we conducted an experiment under which 2 cell types are simulated with 30,000 genes. We randomly selected a small fraction of genes to differ between the 2 cell types while the rest of genes remained the same. As shown in Supplementary Figure 20, SIMLR is able to capture subtle differences between the 2 different cell types with various distributions and scales of noise. SIMLR recovered the two groups of cells with almost 100% accuracy unless the noise is unrealistically large. (NMI = 1 indicates perfectly separable cell types.)

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 7: Details in solving the optimization problem

From Distance to Multiple Kernel Learning

Instead of using a predefined distance metric, we incorporate multiple kernel learning in SIMLR to compute the distances between pairs of cells. The general form of the distance between cell c_i and cell c_j is defined as

$$D(c_i, c_j) = \sum_l w_l D_l(c_i, c_j) = \sum_l w_l \|\phi_l(c_i) - \phi_l(c_j)\|_2^2$$

where $\phi_l(c_i)$ is the l-th kernel-induced implicit mapping of the i-th cell. This mapping is implicit because we are only concerned about the inner products of $\phi_l(c_i)$ and $\phi_l(c_j)$ for pairs (i, j) which is defined as follows:

$$\begin{aligned} \|\phi_l(c_i) - \phi_l(c_j)\|_2^2 &= \phi_l(c_i)^T \phi_l(c_i) + \phi_l(c_j)^T \phi_l(c_j) - 2\phi_l(c_i)^T \phi_l(c_j) \\ &= K_l(c_i, c_i) + K_l(c_j, c_j) - 2K(c_i, c_j) \\ &= 2 - 2K_l(c_i, c_j) \end{aligned}$$

where we only need to compute the kernel functions $K_l(c_i, c_j)$. (The kernel of two identical inputs is set to 1 by convention). In this way, we transform distance between cells using multiple kernel representations.

Details in Updating S

If we fix L and w and treat S as the only optimization variable, the optimization becomes:

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \sum_{i,j} \left[-\sum_l w_l K_l(c_i, c_j) + \gamma (LL^T)_{ij} \right] S_{ij} + \beta \|S\|_F^2 \\ \text{subject to} \quad & \sum_j S_{ij} = 1, S_{ij} \geq 0 \quad \text{for all } (i, j) \end{aligned}$$

Since each row of S is independent in this optimization framework, we can parallel the solution by solving each row, denoted as $\mathbf{s}_1$,...,$\mathbf{s}_N$, independently. Let us denote a N-length vector $\mathbf{v}_i$ (for $i = 1, ..., N$), where

$$(\mathbf{v}_i)_j = -\frac{1}{2\beta} \left(\gamma (LL^T)_{ij} - \sum_l w_l K_l(c_i, c_j) \right).$$

Then the problem can be simplified to N independent quadratic optimization subproblems. For $i = 1..., N$, we can independently solve

$$\begin{aligned} \underset{\mathbf{s}_i}{\text{minimize}} \quad & \frac{1}{2} \|\mathbf{s}_i - \mathbf{v}_i\|_2^2 \\ \text{subject to} \quad & \mathbf{s}_i^T \mathbf{l} = 1, (\mathbf{s}_i)_j \geq 0 \text{ for all } j \end{aligned}$$

where the constraints in these quadratic subproblems are simplex. We can further use the Lagrangian function

$$\mathcal{L}(\mathbf{s}_i, \delta, \sigma) = \frac{1}{2} \|\mathbf{s}_i - \mathbf{v}_i\|_2^2 + \delta(\mathbf{s}_i^T \mathbf{l} - 1) - \sigma^T \mathbf{s}_i,$$

Nature Methods: doi:10.1038/nmeth.4207

where δ is a scaler and σ is a Lagrangian coefficient vector, to set up the following Karush-Kuhn-Tucker conditions (KKT) conditions [33]:

$$\begin{aligned}
(\mathbf{s}_i)_j - (\mathbf{v}_i)_j - \sigma_j + \delta = 0, \qquad & j = 1, 2, \cdots, N \\
(\mathbf{s}_i)_j \geq 0, \qquad & j = 1, 2, \cdots, N \\
\sigma_j \geq 0, \qquad & j = 1, 2, \cdots, N \\
(\mathbf{s}_i)_j \sigma_j = 0, \qquad & j = 1, 2, \cdots, N \\
\mathbf{s}_i^T \mathbf{l} - 1 = 0. &
\end{aligned}$$

By removing the slack variable δ (using standard techniques in [33]), these conditions can be simplified to

$$(\mathbf{s}_i)_j = ((\mathbf{u}_i)_j - \sigma^*)_+, \qquad \sigma_j = (\sigma^* - (\mathbf{u}_i)_j)_+, \quad j = 1, 2, \cdots, N$$

where $x_+ = \max(x, 0)$ and

$$\mathbf{u}_i = (I_N - \frac{\mathbf{l}\mathbf{l}^T}{N})\mathbf{v}_i + \frac{1}{N}\mathbf{l}, \quad \sigma^* = \frac{\mathbf{l}^T \sigma}{N}.$$

Finally, we obtain the solution to the equation system above by solving the following equation:

$$f(\sigma^*) = \frac{1}{N-1} \sum_{j=1}^{N-1} (\sigma^* - (\mathbf{u}_i)_j)_+ - \sigma^* = 0$$

Note that $f(\sigma^*)$ is a piecewise linear and convex function, we use Newton's method to find the root of $f(\sigma^*) = 0$.

Details in Updating L

If we fix S and w and treat L as the only optimization variable, the optimization becomes solving an eigen-system of the matrix $(I_N - S)$. This matrix can be seen as a Laplacian matrix of the similarity graph S. And the solution is obtained by calculating the top C eigenvectors corresponding to the smallest C eigenvalues. Because the task is to simply solve an eigen-system, we used the built-in function `eig()` in MATLAB and `eigen()` in R.

Details in Updating w

Now treating S and F as parameters, the equivalent problem with respect to the only optimization variable w becomes a simple linear programming problem:

$$\begin{aligned}
\underset{w}{\text{maximize}} \quad & \sum_l w_l \sum_{i,j} K_l(c_i, c_j) S_{ij} - \rho \sum_l w_l \log w_l \\
\text{subject to} \quad & w^T \mathbf{l} = 1, \quad w_l \geq 0 \text{ for all } l.
\end{aligned}$$

If the hyper-parameter ρ is set to zero, i.e., $\rho = 0$, then w will collapse to an unit vector where only one kernel is given non-zeros weight in the solution. Otherwise, if $\rho > 0$, then we derive a Lagrangian function to be

$$\mathcal{L}(w) = -\sum_l w_l \sum_{i,j} K_l(c_i, c_j) S_{ij} + \rho \sum_l w_l \log w_l - \delta(w^T \mathbf{l} - 1) - \sigma^T w$$

Nature Methods: doi:10.1038/nmeth.4207

where $\delta > 0$, and $\sigma_l > 0$ for all l. The optimal solution holds when

$$\frac{\partial \mathcal{L}(w)}{\partial w_l} = \sum_{i,j} K_l(c_i, c_j) S_{ij} - \rho(1 + \log w_l) + \delta + \sigma_l = 0.$$

It is easy to see from this equality condition that each w_l is proportional to $\exp\left(\frac{\sum_{i,j} K_l(c_i,c_j)S_{ij}}{\rho}\right)$. Along with the constraint $\sum_l w_l = 1$, we can obtain a closed-form solution for the weights given by

$$w_l = \frac{\exp\left(\frac{\sum_{i,j} K_l(c_i,c_j)S_{ij}}{\rho}\right)}{\sum_l \exp\left(\frac{\sum_{i,j} K_l(c_i,c_j)S_{ij}}{\rho}\right)}.$$

Directly using a closed form expression allows us to compute this update much more efficiently and accurately than using numerical optimization solvers.

Kernel Constructions

Given a feature set that describes a collection of objects, denoted as $X = \{x_1, x_2, \ldots, x_n\}$, we want to construct a similarity kernel $\mathcal{N} \in R^{n \times n}$ for which $\mathcal{K}(i, j)$ indicates the kernel value between the i-th and j-th object. The most widely used method is to assume a Gaussian distribution across pairwise similarities:

$$\mathcal{K}(i, j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right);$$

Here σ is a hyper-parameter that needs careful manual setting. To overcome the large sensitivity to σ, a more advanced method of constructing similarity kernels is proposed in [34]. It estimates the variance using the local scales of the distances. Assume k is the number of neighbors. For the i-th object, e.g, x_i, the associated local variance is estimated as:

$$\mu_i = \frac{\sum_{j \in \mathcal{KNN}(i)} \|x_i - x_j\|}{k}$$

where $\mathcal{KNN}(i)$ denotes all the top k neighbors of the i-th cell. Thus the new kernel is defined as

$$\mathcal{K}_k^\sigma(i, j) = \frac{1}{\sigma(\mu_i + \mu_j)\sqrt{2\pi}} \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2(\mu_i + \mu_j)^2}\right).$$

Different combinations of choices of k and σ would provide us multiple kernels. In all our experiments, we set $k = 10, 12, 14, ..., 30$ and $\sigma = 1.0, 1.25, ..., 2.0$, resulting in 55 different kernels in total.

Convergence Criterion

Because the similarity is solved by SIMLR using an iterative algorithm, it is important to decide a convergence criterion to terminate the iteration. SIMLR adopts an approach similar to spectral clustering [34] to define the convergence criterion. We use the *eigengap* to measure the convergence of our method. For $i = 1, ...N - 1$, the i-th eigengap is defined as follows:

$$eigengap(i) = \lambda_{i+1} - \lambda_i, \tag{1}$$

where λ_i is the i-th eigenvalue of the matrix S and we sort the eigenvalues in an ascending order ($\lambda_1 \leq \lambda_2 \leq \ldots \lambda_N$).

As the number of clusters C is given as parameter to SIMLR, we focus on the value of $eigengap(C) = \lambda_{C+1} - \lambda_C$ in each iteration. The intuition behind using this parameter is that, if a similarity includes C perfectly strong clusters, then $eigengap(C)$ should be near zero (which was proved in [35]). Due to the low-rank constraint in the optimization framework of SIMLR, we would seek a small value of $eigengap(C)$ if SIMLR finds a good optimal value. So a desired stopping criterion for SIMLR is when $eigengap(C)$ is less than a small threshold. However, SIMLR cannot guarantee such small $eigengap(C)$ because of the existence of noise in between-cluster similarities. Therefore, a practical stopping criterion adopted by SIMLR is when $eigengap(C)$ stops decreasing. We empirically observed that the value of $eigengap(C)$ behaves like what we expected within 10 iterations on the four real single-cell data in Supplementary Figure 25.

Details in setting γ and β

We report details in method implementation of SIMLR. It is important to decide the set of hyper-parameters β and γ. We use a data-driven approach to adaptively learn these two hyper-parameters. We use the following rules:

$$\gamma = \beta = \frac{1}{2 * N} \sum_{i=1}^{N} \sum_{j}^{k} (\|x_i - x_i^{k+1}\|^2 - \|x_i - x_i^j\|^2)$$

where x_i denotes the i-th cell, x_i^j denotes the top j-th nearest neighbor of the i-th cell. k is a pre-defined parameter. In this way, we only need to set the number of neighbors we prefer rather than setting two hyper-parameters of β and γ. The number of neighbors is usually easy to set according to the number of cells and locality of the data set. We set $k = 10$ as default for small size data sets and $k = 30$ for relatively larger data set.

The rationality of deciding γ and β using the distance gaps between k-th neighbor and $(k+1)$-th neighbor lies in the fact that, when we solve for S, according to KKT condition, the optimal solution for S is

$$\mathbf{s}_{ij} = (\theta^* - \frac{d_{ij}^2}{2\gamma})_+$$

Therefore, to achieve a desired similarity where top k-neighbor similarities are kept and the rest are set to zeros, we should approximately achieve

$$\frac{d_{i,k_i}^2}{2\gamma} \leq \theta^* \leq \frac{d_{i,(k+1)_i}^2}{2\gamma} \tag{2}$$

where k_i denotes the k-th neighbors of cell i. Due to the constraints that, $\mathbf{s_i}^T \mathbf{l} = 1$, we have

$$k * \theta^* - \sum_{j=1}^{k} \frac{d_{i,k_i}^2}{2\gamma} = 1.$$

This equation leads us to the following observation:

$$k \frac{d_{i,(k+1)_i}^2}{2\gamma} - \sum_{j=1}^{k} \frac{d_{i,k_i}^2}{2\gamma} \geq 1$$

If we set the inequality to equality, we can get an estimation of γ:

$$\gamma \sim \frac{k d^2_{i,(k+1)_i} - \sum_{j=1}^{k} d^2_{i,k_i}}{2}.$$

Similarly, β is set to be equal to γ as follows:

$$\gamma = \beta = \frac{1}{2 * N} \sum_{i=1}^{N} \sum_{j}^{k} (\|x_i - x_i^{k+1}\|^2 - \|x_i - x_i^j\|^2)$$

Since these two parameters control the regularization strength, we adaptively update the parameters during each iteration:

$$\gamma^{(t+1)} = \gamma^{(t)} * (1 + 0.5\delta\{eigengap(C) > 1e-6\})$$

where t denotes the number of iterations. This means that, we increase the value of γ if $eigengap(C)$ is large.

Details in updating S

To provide robustness to SIMLR, we use a smoothed version of updating S:

$$S^{(t+1)} = (1 - \alpha) * S^{(t)} + \alpha * S_+^{(t)}$$

where $S^{(t)}$ is the output of SIMLR after t-th iteration and $S_+^{(t)}$ is the solution by solving Eqn.(5) in the main text. We set $\alpha = 0.8$ as default.

Summary of hyper-parameters in SIMLR

We summarize all the hyper-parameters used in SIMLR:

C: number of desired clusters. This is usually an user-given parameters. It should be data set dependent. We also provide a method to estimate this parameters before applying SIMLR.

B: the number of dimensions of the embedding from the learned similarities by SIMLR. It is usually set to be equal to C.

γ: the weight parameter on the low-rank constraint. It is estimated by the distance gap between $k+1$ and k-th neighbors in a data-driven fashion.

β: the regularization parameters of the norm for S. It is set to be equal to γ in a data-driven fashion.

k: the number of neighbors. This parameter is usually set between $10 \sim 20$ for small scale data sets, and $30 \sim 50$ for large-scale data set.

τ: the smoothing parameters in similarities diffusion. SIMLR is not sensitive to this parameter and usually set a default value of 0.8.

α: the moment parameters in similarity updating. SIMLR is not sensitive to this parameter and usually set a default value of 0.8.

Note that, when we decide neighbors of each cell, we calculate exact nearest neighbors in small scale datasets and using an approximate search for nearest neighbors for large scale datasets.

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 8: Stochastic Neighbor Embedding

The principles we apply below follow from [36] and we incorporated the original framework to improve the embedding for noisy single-cell data.

Let y_i be the coordinates in the dimension that the data is visualized, and $Q_{i,j}$ be the similarity (or joint probability) of cell i and cell j in this dimension, given by

$$Q_{i,j} = \frac{\left(1 + ||y_i - y_j||_2^2\right)^{-1}}{\sum_{k \neq \ell} \left(1 + ||y_k - y_\ell||_2^2\right)^{-1}}. \tag{3}$$

We convert the similarity matrix S into a symmetric joint probability by $P_{i,j} = \frac{1}{N} S_{i,j}$, and minimize the Kullback-Leibler divergence between the two joint probabilities

$$\underset{\mathcal{Y}=(y_1,\ldots,y_N)}{\text{minimize}} KL(P||Q) \tag{4}$$

where

$$KL(P||Q) = \sum_i \sum_j P_{i,j} \log \frac{P_{i,j}}{Q_{i,j}}.$$

This optimization problem can be effectively solved using the gradient descent algorithm with the gradient [36]

$$\frac{\partial}{\partial y_i} KL(P||Q) = 4 \sum_j (P_{i,j} - Q_{i,j})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}.$$

in the update formula

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\partial}{\partial y_i} KL(P||Q) + m(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$$

where $\mathcal{Y}^{(t)}$ are the coordinates of the embedded data and $m(t)$ is the momentum at at iteration t, and η is the learning rate. The second term in the update formula is an additional momentum term to improve convergence.The heuristics and default parameters in t-SNE implementation were also included in SIMLR [36].

Comparison with tSNE

Though bearing some similarity with tSNE, SIMLR has significant novelty in its core advantage of learning similarities. First of all, SIMLR is specifically designed for single-cell data which suffers from large noise and dropout. We use multiple Gaussian kernels and low rank constraints in learned similarities to force cells in different subpopulations to diverge from each other. This is lacking in tSNE, which only applies a t-distribution assumption in similarities. Second, SIMLR is not restricted to any clustering methods on similarities. As we already shown, SIMLR greatly improves the performances of K-means, Affinity Propagation which takes inputs of pair-wise similarities (or distances). Third, SIMLR can also utilize its learned similarity to perform differential gene analysis without even clustering, which tSNE is not capable of. Last but not least, it is well-known that tSNE is not able to perform clustering on large-scale data sets while SIMLR with large-scale extension can easily handle data sets consisting of hundreds of thousands of cells.

Nature Methods: doi:10.1038/nmeth.4207

To show SIMLR is more suitable in single-cell analysis than tSNE, we perform a rigorous comparison with tSNE in various settings. We use Buttner data set as an example here.

First of all, we consider different gene filtering methods: (P1) Gene filtering based on variance; (P2) Gene filtering based on proportion of incorporations. These two gene filtering methods have been widely used in many single-cell analysis. Without any prior knowledge, they seem to be the most reasonable way to reduce the complexity of the input data. P1 screens out genes that are relatively stable across all the cells (i.e., low variance), while P2 removes genes that have few incorporations of cells(i.e, high sparsity across cells). We vary number of genes selected to show how tSNE and SIMLR performs in clustering (See Supplementary Figure 26).

Second, we consider different dimension reduction methods before applying tSNE and SIMLR. (D1) PCA to reduce dimensions first; (D2) SVD to reduce dimensions. These two methods are among the most popular dimension reduction methods. We vary the number of dimension used and report the performances of tSNE and SIMLR (See Supplementary Figure 27).

To conclude, our method outperforms tSNE in various settings of preprocessing of single-cell RNA-seq data.

Supplementary Note 9: Estimation of the Number of Clusters

We provide two intuitive methods to determine the number of desired clusters, C, in SIMLR. The optimal value of C should be close to the true number of subpopulations among the cells.

Eigengap

One direct approach to discover the number of clusters is to analyze the eigenvalues of the weight matrix and searching for a drop in the magnitude of the eigenvalue gaps. Eigengap is defined as follows:

$$eigengap(i) = \lambda_{i+1} - \lambda_i,$$

where λ_i is the i-th eigenvalue of the laplacian matrix and we sort the eigenvalues in an ascending order (i.e., $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.) The best number of clusters C is estimated as :

$$C^* = \max_{i>1} eigegap(i)$$

Separation Cost

We use an alternative approach by analyzing eigenvectors of the Laplacian, similar to [37]. Consider a data set with C disjoint clsuters. It is well known that the eigenvectors of the similarity Laplacian form a full basis spanning the network subspace. Although presence of noise may cause this ideal case to fail, it can still shed light on cluster membership. Given a specific number of clusters C, we aim to find an indication matrix $Z(R) = XR$, where $X \in \mathbb{R}^{n \times C}$ is the matrix of the top eigenvectors of the similarity Laplacian, and $R \in \mathbb{R}^{C \times C}$ is a rotation matrix. Let $[M(R)]_i = \max_j [Z(R)]_{i,j}$. We search for R such that it minimizes the following cost function

$$J(R) = \sum_{i,j} \frac{[Z(R)]_{i,j}^2}{[M(R)]_i^2}$$

We call this objective function as 'separation cost'. Minimizing this cost function over all possible rotations will provide the best alignment with the canonical coordinate system. This is done using the gradient descent scheme [37]. Instead of taking the number of clsuters to be the one providing the minimal cost as in [37], we seek the number of clusters that result in the largest drop in the value of $J(R)$.

Nature Methods: doi:10.1038/nmeth.4207

Supplementary Note 10: Details in Large Scale Extension of SIMLR

We decompose large-scale SIMLR into three steps: 1) Learning SIMLR using KNN graph approximation; 2) Clustering cells from learned similarity; 3) Visualization from the learned similarity. We discuss implementation details and perform complexity analysis in each of these steps.

Similarity learning in SIMLR

In large scale data set, fully pair-wise similarity is prohibitive not only in computation time but also in memory. Therefore, we adopt K Nearest Neighbor (KNN) approximation of the full similarities, in which, for each cell, only top K approximate nearest neighbors (ANN) have non-zero similarities. This will not only reduce memory consumption but also greatly improve the speed in our optimization solving. We use a fast algorithm named Annoy [1]. Annoy is an empirically engineered algorithm that has been recognized as one of the best nearest neighbor search libraries. Note that our SIMLR is not restrictive about the choice of nearest neighbor search libraries. For example, empirical results using another ANN library named kGraph [2] also gives us similar competitive results. Further, in practical implementations, we first apply a fast version of randomized PCA [38] before searching nearest neighbors. This will greatly improve the speed of ANN search.

Our optimization framework solves for S, L and w iteratively. When solving S, since we only updates similarities for K neighbors, the time complexity is $O(KN)$. When solving L, we need to perform an eigen-decomposition of the Laplacian matrix, which is also sparse. This operation is the most time-consuming among all the other operations in SIMLR. However, this operation can be greatly speeded up in SIMLR since we only use sparse similarities [39]. We adopt a well-known package named Spectra [3] to estimate the top eigenvectors. Empirical experiments suggests that it only takes less than one minute to obtain eigenvectors for a sparse similarities between tens of thousands of cells.

As for similarity diffusion for large scale data sets, it is computational prohibitive to calculate the closed form of diffusion since it is of complexity $O(N^2)$, where N is the number of cells. However, since we only keep top K neighbors, we only need to update similarities for these pre-selected neighbors. Using the updating rules of diffusion , we only apply five steps of diffusions. Note that, the time complexities in our simplified diffusion is $O(TKN)$, where T is the number of iterations, K is the number of neighbors.

Clustering on learned similarities

Once we obtain cell-to-cell similarities, we aim to perform clustering to dissect the heterogeneity among cells. On small-scale data sets, we can perform tSNE-type embeddings to get a low-dimensional embedding which preserves the learned similarities. However, this is computationally prohibitive on large-scale data sets. Therefore, we propose to perform a simple spectral clustering method [35] directly based on the learned similarities without any explicit embeddings. In fact, the way we solve for L inside our optimization framework provides us a direct way to perform spectral clustering by running k-means on the final solution of L. To increase robustness of the

[1] https://github.com/spotify/annoy
[2] http://www.kgraph.org/
[3] http://yixuan.cos.name/spectra/

Nature Methods: doi:10.1038/nmeth.4207

final clustering result, we run k-means many times (e.g, 50) and take the one with optimal objective value [22].

Visualization from learned similarities

Visualization is another essential application of SIMLR. Once we obtain the similarities by SIMLR, we want to map all the cells to 2D space based on the similarities. To overcome the large computational and memory burden in tSNE embeddings, we adopt Barnes-Hut approximation [40] to estimate the gradient of the tSNE objective. Barnes-Hut technique splits the 2D space into four regions by constructing a quad tree and each sub-region can be further divided into four regions recursively. The splitting rule is that, if one region contains more than one data point, it is divided to four smaller bounding boxes. Therefore, it is easy to see that, the memory complexity is $O(2^d * h)$ where d is the dimensions of the low embeddings (in the case of visualization, $d = 2$) and h is the depth of the quad tree. The gradient of the embedding objective is calculated as follows:

$$\frac{\partial \mathcal{C}}{\partial \mathbf{y}_i} \propto \sum_{\mathcal{T}} Z(\mathbf{y}_i - \mathbf{y}_{\mathcal{T}})$$

where $\mathcal{T}$ denotes a tree in Barnes-Hut approximation, $Z(\cdot)$ represents the detailed differential functions for the distribution of the low-dimensional embeddings(e.g, in tSNE-type embeddings, it is the first-order gradient of t-student functions). Clearly, the computational complexity in gradient calculation is $O(N \log N)$. We refer the details of Barnes-Hut approximation to [40].

However, the complexity can be further reduced due to the structures of sparse similarities by SIMLR. When calculating gradients, instead of adding all the trees, we only consider trees that contain the neighbors of each specific cell:

$$\frac{\partial \mathcal{C}}{\partial \mathbf{y}_i} \propto \sum_{\mathcal{T}_i} Z(\mathbf{y}_i - \mathbf{y}_{\mathcal{T}_i})$$

where $\mathcal{T}_i$ denotes trees containing at least one of the neighbors of cell i. In this case, we reduce the complexity of $O(\log(N)NK)$, where K is the number of neighbors.

Nature Methods: doi:10.1038/nmeth.4207

References

[1] Damian Szklarczyk, Andrea Franceschini, Michael Kuhn, Milan Simonovic, Alexander Roth, Pablo Minguez, Tobias Doerks, Manuel Stark, Jean Muller, Peer Bork, et al. The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored. *Nucleic acids research*, 39(suppl 1):D561–D568, 2011.

[2] Florian Buettner, Kedar N Natarajan, F Paolo Casale, Valentina Proserpio, Antonio Scialdone, Fabian J Theis, Sarah A Teichmann, John C Marioni, and Oliver Stegle. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature biotechnology*, 33(2):155–160, 2015.

[3] Jüri Reimand, Meelis Kull, Hedi Peterson, Jaanus Hansen, and Jaak Vilo. g:Profiler - a web-based toolset for functional profiling of gene lists from large-scale experiments. *Nucleic acids research*, 35(suppl 2):W193–W200, 2007.

[4] Daniele Merico, Ruth Isserlin, Oliver Stueker, Andrew Emili, and Gary D Bader. Enrichment map: a network-based method for gene-set enrichment visualization and interpretation. *PloS one*, 5(11):e13984, 2010.

[5] Alex A Pollen, Tomasz J Nowakowski, Joe Shuga, Xiaohui Wang, Anne A Leyrat, Jan H Lui, Nianzhen Li, Lukasz Szpankowski, Brian Fowler, Peilin Chen, et al. Low-coverage single-cell mrna sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex. *Nature biotechnology*, 2014.

[6] Aleksandra A Kolodziejczyk, Jong Kyoung Kim, Jason CH Tsang, Tomislav Ilicic, Johan Henriksson, Kedar N Natarajan, Alex C Tuck, Xuefei Gao, Marc Bühler, Pentao Liu, et al. Single Cell RNA-Sequencing of Pluripotent States Unlocks Modular Transcriptional Variation. *Cell stem cell*, 17(4):471–485, 2015.

[7] Dmitry Usoskin, Alessandro Furlan, Saiful Islam, Hind Abdo, Peter Lönnerberg, Daohua Lou, Jens Hjerling-Leffler, Jesper Haeggström, Olga Kharchenko, Peter V Kharchenko, et al. Unbiased classification of sensory neuron types by large-scale single-cell RNA sequencing. *Nature neuroscience*, 18(1):145–153, 2015.

[8] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 5:401–409, 1969.

[9] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[10] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[11] Charles Spearman. " General Intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):201–292, 1904.

[12] Michael E Tipping and Chris M Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999.

[13] Martin A Turman, Toshio Yabe, Cynthia McSherry, Fritz H Bach, and Jeffrey P Houchins. Characterization of a novel gene (NKG7) on human chromosome 19 that is expressed in natural killer cells and t cells. *Human immunology*, 36(1):34–40, 1993.

[14] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

[15] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, December 2010.

[16] László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.

[17] Silke Wagner and Dorothea Wagner. *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

[18] A Vedaldi, B Fulkerson, and VL Feat. An open and portable library of computer vision algorithms. In *Proceedings of the 18th annual ACM international conference on Multimedia*, pages 1469–1472, 2007.

[19] Emma Pierson and Christopher Yau. ZIFA: dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome biology*, 16(1):1, 2015.

[20] Christopher M Bishop. Pattern recognition. 2006.

[21] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[22] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29. ACM, 2004.

[23] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[24] David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–144, 2006.

[25] Grace X.Y. Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Donald A Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H Bielas. Massively parallel digital transcriptional profiling of single cells. *bioRxiv*, 2016.

[26] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[27] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[28] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[29] Peiguo G Chu and Daniel A Arber. CD79: a review. *Applied Immunohistochemistry & Molecular Morphology*, 9(2):97–106, 2001.

[30] Luc Van Kaer. Regulation of immune responses by CD1d-restricted natural killer T cells. *Immunologic research*, 30(2):139–153, 2004.

[31] Laleh Haghverdi, Florian Buettner, and Fabian J Theis. Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics*, 31(18):2989–2998, 2015.

[32] El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe'er. visne enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature biotechnology*, 31(6):545–552, 2013.

[33] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[34] Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature methods*, 11(3):333–337, 2014.

[35] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[36] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[37] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2005.

[38] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2009.

[39] Bo Wang, Junjie Zhu, Armin Pourshafeie, Oana Ursu, Serafim Batzoglou, and Anshul Kundaje. Unsupervised learning from noisy networks with applications to hi-c data. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3297–3305. Curran Associates, Inc., 2016.

[40] Laurens Van Der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

Nature Methods: doi:10.1038/nmeth.4207