


RESEARCH

Open Access

Tuning parameters of dimensionality reduction methods for single-cell RNA-seq analysis

Felix Raimundo¹, Celine Vallot^{2,3*} and Jean-Philippe Vert^{1*} 

*Correspondence:

celine.vallot@curie.fr;

jpvert@google.com

²CNRS UMR3244, Institut Curie, PSL
Research University, 75005 Paris,
France

³Translational Research
Department, Institut Curie, PSL
Research University, 75005 Paris,
France

Full list of author information is
available at the end of the article

Abstract

Background: Many computational methods have been developed recently to analyze single-cell RNA-seq (scRNA-seq) data. Several benchmark studies have compared these methods on their ability for dimensionality reduction, clustering, or differential analysis, often relying on default parameters. Yet, given the biological diversity of scRNA-seq datasets, parameter tuning might be essential for the optimal usage of methods, and determining how to tune parameters remains an unmet need.

Results: Here, we propose a benchmark to assess the performance of five methods, systematically varying their tunable parameters, for dimension reduction of scRNA-seq data, a common first step to many downstream applications such as cell type identification or trajectory inference. We run a total of 1.5 million experiments to assess the influence of parameter changes on the performance of each method, and propose two strategies to automatically tune parameters for methods that need it.

Conclusions: We find that principal component analysis (PCA)-based methods like scran and Seurat are competitive with default parameters but do not benefit much from parameter tuning, while more complex models like ZinbWave, DCA, and scVI can reach better performance but after parameter tuning.

Introduction

Single-cell RNA sequencing (scRNA-seq) is a powerful technology to characterize the transcriptomic profile of individual cells within a population [1]. By allowing researchers to identify cell types based on their transcriptomic signatures instead of pre-defined markers, it is rapidly establishing itself as a standard tool to answer a variety of biological questions, ranging from characterizing the heterogeneity of complex tissues [2, 3] to discovering new cell types [4] or elucidating cell differentiation processes [5].

The analysis of scRNA-seq data raises, however, a number of challenges. Due to the small amount of RNA available in each individual cell, and to the technical difficulty to analyze thousands (or millions) of cells in parallel, raw scRNA-seq data have been found to



© The Author(s). 2020 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

be subject to a number of biases including low sequencing depth, over-dispersion and zero inflation of read counts, or sensitivity to batch effects [6, 7]. Many computational methods have therefore been developed in recent years to take into account the specificities of scRNA-seq data and address the issues of data normalization, cell type identification, differential gene expression analysis, cell hierarchy reconstruction, or gene regulatory network inference (see [8, 9] for recent reviews). In order to help practitioners choose an analysis pipeline among the many available, several studies have benchmarked algorithms and softwares for applications such as dimensionality reduction [10], clustering [11, 12], differential expression [13], or trajectory inference [14].

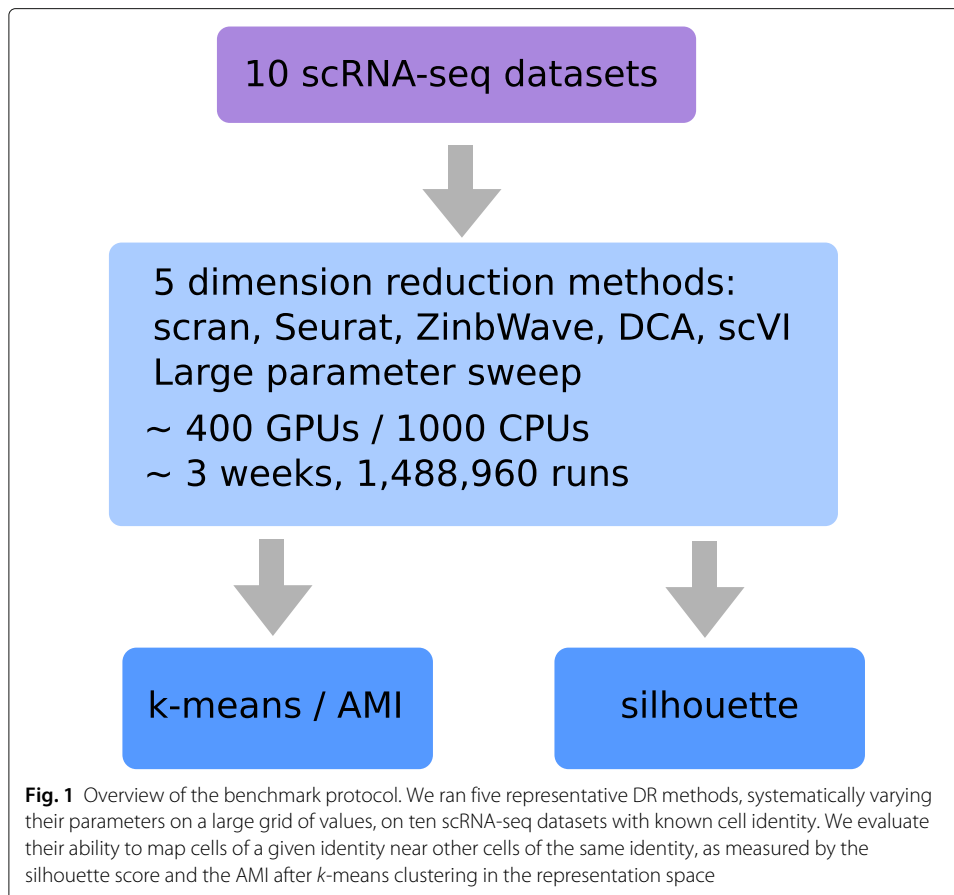
One shared caveat by these benchmarking efforts, however, is that the methods tested are run with their default parameters. This may not reflect what an educated user would do in practice and does not address the practical questions of (i) whether parameter tuning is relevant at all for a given method and (ii) how to tune parameters if needed. Recently, [15] highlighted the relevance of these questions, showing that variation autoencoders (VAE) algorithms for dimension reduction (DR) of scRNA-seq work very well once properly tuned, but are extremely sensitive to changes in parameters and can dramatically fail if not properly tuned.

Here, we propose to challenge this issue of parameter tuning focusing on methods for DR of scRNA-seq data, not only because they can be directly useful for visualization purpose, but also because DR is a common first step for most downstream applications such as cell type identification or trajectory inference [8, 9]. We propose a new benchmark protocol for DR methods, composed of ten scRNA-seq datasets of various complexity mixing experimentally characterized populations of cells, where we measure the quality of a DR method by its ability to map cells of a given cell type near each other in the representation space. Using this protocol, we benchmark five popular and representative DR methods, combining both PCA-based methods, a matrix factorization method, and VAE methods, systematically varying their tunable parameters. The resulting ~ 1.5 million experiments reveal not only the performance of DR methods using their default parameters, but also their performance if parameters are properly tuned. We find in particular that principal component analysis (PCA)-based methods like scran [16] and Seurat [17] are competitive with default parameters but do not benefit much from parameter tuning, while more complex models like ZinbWave [18], DCA [19], and scVI [20] can reach better performance but after parameter tuning. We propose and evaluate two strategies to tune parameters automatically, either by changing the default parameters or by optimizing a heuristic on each new dataset. In spite of promising results for some of the methods like ZinbWave, both strategies sometimes identify very suboptimal parameters, suggesting that parameter tuning for complex DR models on dataset without ground truth annotation remains an important but largely open problem.

Results

A benchmark of DR methods for scRNA-seq data

A DR method takes a scRNA-seq dataset as input and maps each individual cell to a point in d -dimensional *representation space*, where downstream applications such as cell type prediction or lineage reconstruction are performed. In order to empirically assess the quality of DR methods and the influence of parameter tuning, we propose a benchmark protocol, summarized in Fig. 1, where we collected ten diverse scRNA-seq datasets with



experimentally validated cell types, and evaluate five representative DR methods, tested on a large parameter sweep, according to their ability to map cells of a given origin near to other cells of the same cell type.

Table 1 and Fig. 2 summarize the main features of the ten datasets.

Each dataset contains hundreds to thousands of experimentally characterized cell types, either derived from known cell lines or purified by FACS. The ten datasets vary in the technology used (10x, CEL-Seq2 or Smart-Seq2), the organism of origin (human or murine), and the overall biological complexity of the mixture. More precisely, the first five datasets (Zhengmix4eq to Zhengmixun8eq) are in silico mixtures of FACS purified human immune cell populations from [21] produced with 10x, comprising either equal mixes of four, five, and eight cell populations or unequal mixes of four and eight cell populations. The datasets with five and eight cell populations are particularly challenging, since they both contain five closely related T cell populations. The next four datasets (sc_10x to sc_celseq2_5cl) are in vivo mixtures of three or five human cell lines from [12], sequenced by CEL-Seq2 or 10x. Finally, the last dataset is an in silico mixture of four FACS purified mouse tissues from [22] produced with Smart-Seq2, where we selected tissues with no overlap in cell types.

We use this benchmark to evaluate the performance of five popular computational pipelines for DR of scRNA-seq data: scran [16], Seurat [17], ZinbWave [18], DCA [19], and scVI [20]. These pipelines are all publicly available as R or Python packages and can

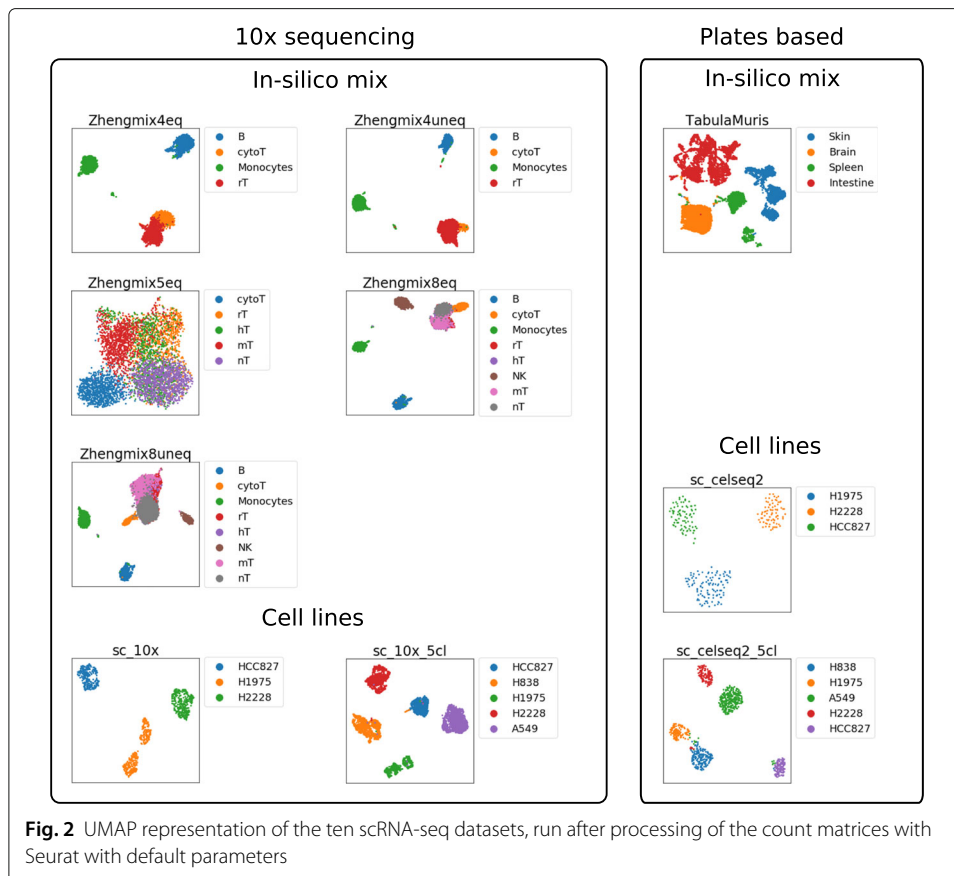
Table 1 Benchmark datasets

Dataset	Technology	Organism	N cells	Cell types	N types	Ref.
Zhengmix4eq	10x	Human	3,909	B (24.5%), Monocytes (24.5%), cytoT (25.5%), rT (25.5%)	4	[11, 21]
Zhengmix4uneq	10x	Human	6,345	B (15%), Monocytes (30%), cytoT (8%), rT (46%)	4	[11, 21]
Zhengmix5eq	10x	Human	4,876	hT (20%), mT (20%), cytoT (20%), nT (20%), rT (20%)	5	[21]
Zhengmix8eq	10x	Human	3,908	B (12.5%), Monocytes (14.5%), hT (10%), NK (15%), mT (12.5%), cytoT (10%), nT (13%), rT (12.5%)	8	[11, 21]
Zhengmix8uneq	10x	Human	6,350	B (7.5%), Monocytes (15%), hT (8%), NK (4%), mT (15%), cytoT (4%), nT (23%), rT	8	[21]
sc_10x	10x	Human	902	H1975 (34.5%), H2228 (35%), HCC827 (30.5%)	3	[12]
sc_10x_5cl	10x	Human	3,918	A549 (32%), H1975 (11%), H2228 (19.5%), H838 (22.5%), HCC827 (15%)	5	[12]
sc_celseq2	CEL-Seq2	Human	274	H1975 (41%), H2228 (29.5%), HCC827 (29.5%)	3	[12]
sc_celseq2_5cl	CEL-Seq2	Human	895	A549 (36%), H1975 (14.5%), H2228 (14%), H838 (22%), HCC827 (13.5%)	5	[12]
TabulaMuris	Smart-Seq2	Murine	12,081	Brain (36.5%), intestine (31.5%), skin (19%), spleen (13%)	4	[22]

The first five datasets are derived from [21] and [11] and contain CD19+ B cells (B), CD14+ monocytes (Monocytes), CD4+ helper T cells (hT), CD56+ natural killer cells (NK), CD4+/CD45RO+ memory T cells (mT), CD8+/CD45RA+ naive cytotoxic T cells (cytoT), CD4+/CD45RA+/CD25- naive T cells (nT), and CD4+/CD25+ regulatory T cells (rT). The four next are from [12] and contain the five following cell lines: A549, H1975, H2228, H838, and HCC827. The last one is from [22]

process datasets containing thousands of cells in a reasonable amount of time (less than 12 h on a GPU/CPU with 10 cores). They all implement processing steps specific to scRNA-seq data together with representative DR methods including principal component analysis (PCA) for scran and Seurat, matrix factorization for ZinbWave, and (variational) autoencoders for DCA and scVI. While these pipelines also implement various downstream tasks such as cell clustering or differential expression analysis, we restrict our analysis to the DR step.

To quantify the ability of a DR method to map biologically similar cells to similar locations in the representation space, we use two complementary measures: the silhouette, on the one hand, and the adjusted mutual information (AMI) when the cells are clustered with the k -means algorithm, on the other hand (see details in the “[Material and methods](#)” section). Both measures vary between 0 for a random embedding to 1 for an embedding that perfectly preserves the cell type information. Silhouette is a measure agnostic to any particular clustering algorithm, and measures how close a cell is to other cells of the same type compared to cells of different types; for the silhouette to be large, cell types must not only be separated, but also form compact clusters far from each other. AMI, on the other hand, directly measures how well a particular clustering algorithm recovers known cell types and is therefore a good proxy for the performance of cell type identification as a downstream task of DR. Both measures are frequently used to assess the performance of cell embedding techniques [10, 11, 18, 20, 23]. Another standard measure to assess the



ability of a clustering algorithm to recover known classes is the adjusted rand index (ARI) [10, 11, 20, 23]; however, we found that AMI and ARI are extremely correlated (Additional file 1: Figure S1) and lead to similar conclusions, so we just report results based on AMI below.

Performance of five popular DR methods with default parameters

We first assess the performance of each method with its default parameters, except for the dimension of the representation space which we arbitrarily set to 10 for all methods. Indeed, the performance scores (AMI and silhouette) strongly vary with the dimension (Additional file 1: Figure S2 and S3), so fixing the dimension allows to compare more fairly the different DR methods. Figure 3a (with “default” legend) shows the performance reached by each method on each dataset, in terms of AMI (left) and silhouette (right), and Table 2 summarizes the mean performance reached by each method over the datasets.

As expected, Fig. 3a clearly shows that for all methods, the performance varies across datasets, in a rather consistent manner. In terms of AMI, the four cell lines datasets tend to be the easiest (AMI > 0.9 for most methods), followed by TabulaMuris and the two Zhengmix mixtures of four cell lines (AMI in the range 0.7~0.9 for most methods), followed by the three Zhengmix mixtures containing the five closely related T cell populations (AMI in the range 0.1~ 0.7 for most methods). The silhouette scores overall follow the same trend, although the difference between the first two groups of datasets is less pronounced. Interestingly, we see that in cases where several DR methods allow to almost perfectly

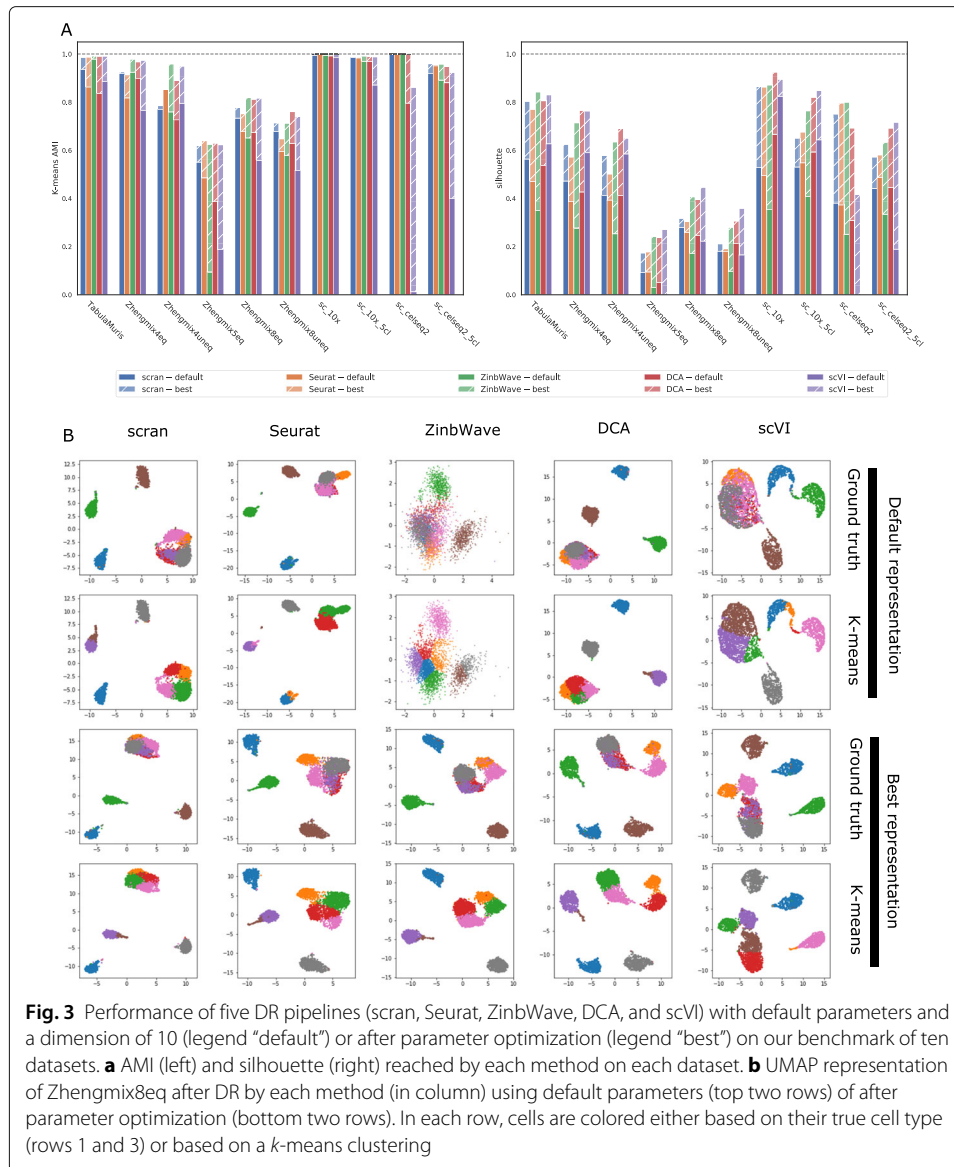


Fig. 3 Performance of five DR pipelines (scran, Seurat, ZinbWave, DCA, and scVI) with default parameters and a dimension of 10 (legend “default”) or after parameter optimization (legend “best”) on our benchmark of ten datasets. **a** AMI (left) and silhouette (right) reached by each method on each dataset. **b** UMAP representation of Zhengmix8eq after DR by each method (in column) using default parameters (top two rows) or after parameter optimization (bottom two rows). In each row, cells are colored either based on their true cell type (rows 1 and 3) or based on a *k*-means clustering

Table 2 Mean performance on the ten datasets of each method in terms of AMI and silhouette

Method	Mean AMI				Mean silhouette			
	Default	Best	ANOVA AMI heuristic	Silhouette heuristic	Default	Best	ANOVA AMI heuristic	Silhouette heuristic
scran	0.840	0.868	0.841	0.741	0.362	0.547	0.396	0.494
Seurat	0.788	0.860	0.814	0.683	0.369	0.543	0.490	0.373
ZinbWave	0.780	0.896	0.851	0.825	0.249	0.609	0.562	0.591
DCA	0.758	0.885	0.837	0.583	0.396	0.639	0.403	0.381
scVI	0.560	0.872	0.709	0.510	0.384	0.621	0.482	0.318

The “Default” columns correspond to the performance of each method using its default parameters, with a dimension of 10. The “Best” column corresponds to the best performance reached after varying the parameters. The “ANOVA AMI heuristic” column corresponds to the performances of the new default parameters described in the “[Influence of parameters on performance](#)” section. The “Silhouette heuristic” column corresponds to the performance of the heuristic described in “[Tuning parameters in practice](#)” section

cluster the cell types, such as sc_10x or sc_10x_5cl, the silhouette is usually far from 1 and varies across methods, illustrating the complementarity of both measures.

Besides variations across datasets, we also observe variations across DR methods. As shown in Table 2, scran has the best AMI on average (mean AMI = 0.84), followed by Seurat, ZinbWave, and DCA (mean AMI = 0.75~0.79), but this ranking is not statistically significant (p value > 0.05 for Wilcoxon one-way test), while scVI is clearly behind (mean AMI = 0.56) (p value < 0.05 for all methods). As suggested in Fig. 3b on Zhengmix8eq, for example, scVI with default parameters does not manage to clearly isolate the three non-T cell clusters, resulting in errors in k -means clustering. In terms of silhouette, all methods are very similar (mean silhouette = 0.36~0.39), except for ZinbWave which is clearly behind (p value < 0.05 for all methods). The reason why ZinbWave tends to have a correct AMI but a poor silhouette is suggested by Fig. 3b, where we see on Zhengmix8eq that ZinbWave (with default parameters) produces a representation good enough for k -means to correctly recover most of the cell types, but where the different clusters look much less compact and separated from each other than with other DR methods. The p values for the comparisons can be found in Additional file 1: Figure S16–S17.

While these relative performances hold for a representation in dimension 10, the performance of some methods fluctuates with the dimension of the embedding space (Additional file 1: Figure S2 and S3). While scran and Seurat are rather insensitive to increase in dimension after 10, DCA's mean AMI tends to increase in higher dimensions, while ZinbWave's and scVI's mean AMI decrease with the dimension, suggesting that different methods need more or less dimension to capture the same biology.

This average performance of DR methods hides important variations across datasets, as visualized in Fig. 3a. For example, we see that scVI has specifically poor performance compared to other methods on the two CEL-Seq2 datasets, which may be due to a particularity of this technology or to the fact that both datasets (sc_celseq2 and sc_celseq2_5cl) have a relatively small number of cells. The difference in AMI across methods, and the good behavior of the linear models underlying scran, Seurat and ZinbWave, is most visible on the “difficult” Zhengmix mixtures containing the five closely related T cell populations, while the difference in silhouette, and the good behavior of the nonlinear models underlying DCA and scVI, is more visible in the “easy” datasets where all methods have an AMI above 0.7.

Performance reachable across a parameter sweep

While using a computational pipeline with default parameters is often the method of choice for practitioners, there is little guarantee that default parameters are adapted to all situations. In particular, the performance of deep learning-based methods for scRNA-seq analysis was shown to be highly sensitive to choices of parameters [15]. In order to assess the performance of each DR method if all parameters were properly tuned in a dataset-specific way, we now run each method by sweeping all tunable parameters across a large grid of values, as summarized in Table 3, and compute the performance reached by each method after cherry picking a posteriori the best parameters. Note that the resulting performance is therefore an upper bound on the performance that each method can reach if parameters are tuned without knowing the ground truth.

In total, sweeping across the grid of parameters results in 384 different runs per dataset for scran and ZinbWave, 288 for Seurat, 40,320 for scVI, and 107,520 for DCA, hence

Table 3 Description of parameter sweep

Method	Parameters	Values
scran	Size factors normalization	{ True , False }
	ERCC counts normalization	{ True , False }
	Assay type	{ logcounts , counts }
	High variance genes	{ 100, 300, 500 , 1000, 2000, 3000 }
	Dimension of latent space	{ 2, 8, 10, 16, 32, 50 , 64, 128 }
Seurat	Normalization method	{ LogNormalize , CLR }
	Criteria for high variance genes	{ vst , mvp, dist }
	High variance genes	{ 100, 300, 500, 1000, 2000 , 3000 }
	Dimension of latent space	{ 2, 8, 10, 16, 32, 50 , 64, 128 }
ZinbWave	Gene covariates	{ True, False }
	Epsilon (regularizer)	{ 200, 500, 1000 , 2000 }
	High variance genes	{ 100 , 300, 500, 1000, 2000, 3000 }
	Dimension of latent space	{ 2 , 8, 10, 16, 32, 50, 64, 128 }
DCA	Dispersion and reconstruction	{ zinb-conddisp , zinb, nb-conddisp, nb }
	Batch normalization	{ True , False }
	Dimension of the latent space	{ 2, 8, 10, 16, 32 , 50, 64, 128 }
	Number of training epochs	{ 20, 50, 100, 200, 300 , 500, 1000 }
	Normalize counts	{ True , False }
	Scale variance	{ True , False }
	Log normalization	{ True , False }
	Dropout rate	{ 0 , 0.1 }
	Number of hidden neurons	{ 64 , 128, 256 }
	Random seed	{ 0 , 1, 2, 3, 4 }
scVI	Number of hidden neurons	{ 64, 128 , 256 }
	Number of training epochs	{ 20 , 50, 100, 200, 300, 500, 1000 }
	Learning rate	{ 1e-2, 1e-3 , 1e-4 }
	Dropout rate	{ 0, 0.1 }
	Layers	{ 1 , 2 }
	Dimension of the latent space	{ 2, 8, 10 , 16, 32, 50, 64, 128 }
	Dispersion	{ gene , gene-cell }
	Reconstruction loss	{ nb, zinb }
Random seed	{ 0 , 1, 2, 3, 4 }	

For each method (first column), we vary a number of tuneable parameters (second column) systematically over a grid of values (third column). The bold value in the third column is the default value

a total of 1,488,960 DR experiments. Running all experiments took several weeks on a dedicated cluster of 1000 CPUs and 400 GPUs. Out of these runs, 60% ran correctly for scran, 99.97% for Seurat, 96.51% for ZinbWave, 97.96% for DCA, and 100% for scVI. The low number of runs for scran is mostly due to the absence of ERCC in all the 10X datasets and because the size factor computation on TabulaMuris failed. The failures for ZinbWave and DCA were due to memory issues (either of the GPU or CPU). There was a single failure for Seurat whose cause has not been identified.

We report in Fig. 3a and Table 2 (with “Best” legend) the best value reached across the parameter sweep on each dataset, in addition to the performance reached with the default parameters. Overall, we see that for all methods, a gain can result from parameter tuning compared to using default parameters. For AMI, the mean gain across datasets ranges from 0.311 for scVI to 0.028 for scran, while for silhouette, it ranges from 0.185 for scran to

0.360 for ZinbWave. Seurat and scran are the methods that benefit least from parameter tuning, suggesting that default parameters are already good choices across most datasets. Autoencoder-based DCA and scVI benefit more for parameter tuning, and outperform scran and Seurat in mean AMI after parameter tuning, confirming the importance of parameter tuning for these models [15]. Since the number of parameters tested for these models is also two orders of magnitude larger than for Seurat, scran, and ZinbWave, the “best” performance after cherry-picking the best parameters may be over-optimistic for DCA and scVI. As for ZinbWave, a good choice of parameters leads to the best mean AMI across methods (0.896) and the largest improvement in silhouette compared to default parameters.

More precisely, for all cell line datasets and for TabulaMuris, parameter tuning allows all method to reach an almost perfect AMI, including methods like scVI that have a very poor performance with default parameters on *sc_celseq2* and *sc_celseq2_5cl*. On the same datasets, parameter tuning brings an important improvement to the silhouette score of 0.2 to 0.6 to all methods. After parameter tuning, both encoder-based methods (DCA and scVI) tend to outperform ZinbWave, which tends to outperform both PCA-based methods scran and Seurat in terms of silhouette. This highlights the possibilities of nonlinear DR methods to perform DR even on simple datasets, but the need to correctly tune parameters in order to reveal their full potential.

On the immune cell datasets, we see again that tuning parameters allows to boost performance and bridge important gaps between methods in terms of silhouette and that after parameter optimization, both autoencoder-based methods slightly outperform ZinbWave, which slightly outperforms both PCA-based methods. The AMI performance of all methods is also improved by parameter optimization for all methods but scran, and we see no clear and consistent winner after parameter optimization. This suggests that simple PCA-based methods like scran, even with default parameters, are good enough to match the performance of more complex models after parameter tuning in terms of AMI; however, the better silhouette of more complex models once tuned may be an advantage for other downstream applications beyond clustering by *k*-means. The benefits of parameter tuning is further illustrated in Fig. 3b, which shows a UMAP visualization of the representation space learned by the different methods with the default parameters (bottom two rows) or after parameter tuning for AMI (top two rows), for the Zhengmix8eq dataset. For ZinbWave and scVI, which strongly benefit from parameter tuning in this case, we see that the dataset looks very different with the default parameters or after parameter optimization, the different cell types being but better separated in the later case.

Regarding the absolute performance reached across the datasets, it is interesting to note that the best AMI scores for Zhengmix5eq and Zhengmix8uneq (the two hardest datasets) are only between 0.6 and 0.7, when they are above 0.9 for Zhengmix4eq (the easiest one). This suggests that if current DR methods are good at identifying sufficiently different cell types, they have difficulties to differentiate very similar cell types, a variety of T cells in our case, even after parameter optimization.

To check how the AMI results are influenced by the choice of the clustering algorithm (here *k*-means), we also computed the AMI performance when cells are clustered by Ward's hierarchical clustering (HC) or the Louvain algorithm [24]. As shown on Additional file 1: Figure S4, the performance barely changes when *k*-means is replaced by HC.

When Louvain clustering is used, the overall performance decreases on most datasets, possibly because we used default parameters for this algorithm, while k -means and HC are run knowing in advance the correct number of clusters. The relative ordering of the different method is overall conserved, suggesting our conclusions based on AMI are robust to the clustering method.

Influence of parameters on performance

Having shown that parameter tuning has the potential to boost the performance of all methods for all datasets compared to using default parameters, we now investigate in more details the influence of each parameter on the performance. For that purpose, we estimate the mean contribution of each parameter value on the performance (AMI or silhouette) with a factorial analysis of variance (ANOVA, see Methods) procedure. We find that all parameters of all methods, except for “gene covariate” for ZinbWave on the silhouette, have a significant influence on both AMI and silhouette (ANOVA p value < 0.05) and summarize in Additional file 1: Table S1 and S2 the potential effect of tuning each parameter by comparing the best and worse contributions to performance among the values it can take. We see that some parameters can have a very important effect, such as proper log normalization in scran which on average can boost the AMI by 0.50, or the choice of dimension in the latent space for ZinbWave that can boost the silhouette by 0.56 on average. If we arbitrarily define a parameter as “influential” if its potential effect is more than 0.05 on AMI or silhouette, we see that in addition to the dimension of the representation space which is influential for all methods, scran, Seurat, and ZinbWave have one influential parameters (log normalization for scran; normalization method for Seurat; number of top genes for ZinbWave), DCA has two (batch normalization and normalize counts), and scVI has three (dispersion, number of training epochs, and learning rate), suggesting that more care in parameter tuning is needed for the autoencoder-based methods than for the matrix factorization-based methods.

As shown in Table 3 and Additional file 1: Table S1, the best parameter values in terms of mean contribution to the performance are not always the default parameters of each method. This suggests that the best parameter values identified by our analysis, which we refer to below as “ANOVA AMI heuristic” when we pick the parameter values that have the largest positive influence on AMI, may be interesting to use as new default parameters for each method. Note that, by construction, the ANOVA AMI heuristic parameters are the same for all datasets. To test this hypothesis, we report the performance of each DR method using the ANOVA AMI heuristic as default parameters in Additional file 1: Figure S5 and summarize the mean performance across datasets in Table 2.

We see that, on average, all methods benefit from the ANOVA AMI heuristic compared to the existing default parameters, particularly scVI, DCA, and ZinbWave in terms of AMI, and particularly ZinbWave, Seurat, and scVI in terms of silhouette. In particular, ZinbWave outperforms all other methods, both in AMI and in silhouette, with the ANOVA AMI heuristic. Interestingly, we see in Additional file 1: Figure S5 that for all methods, the AMI increases on almost all datasets with the ANOVA AMI heuristic. Of course, these promising results should be taken with care, given that we evaluate the performance of the ANOVA AMI heuristic on the datasets used to perform the ANOVA, but they suggest a systematic approach for method developers to set default parameters.

We now investigate in more details to what extent further performance gain may result from parameter tuning on each dataset separately, as opposed to setting new default parameter values common to all datasets. For that purpose, we estimate again the contribution of each parameter in the final AMI and silhouette, allowing a different contribution in different datasets by adding interaction terms in the factorial ANOVA model between the dataset, on the one hand, and the tunable parameters, on the other hand (see Methods). Note that we remove from this analysis a few parameters that were not tested on all datasets: ERCC for scran on 10x datasets, and sum factor normalization for scran on TabulaMuris. Additional file 1: Figure S6–S15 summarize the contributions of each parameter in each dataset for AMI and silhouette, as estimated from the factorial ANOVA with interactions. All interactions between dataset and parameters are significantly non zero (p value < 0.05), except for the interaction between dataset and the “gene covariate” parameter of ZinbWave, showing that the influence of most parameters on the final performance is not the same across datasets. To assess whether dataset-specific parameter tuning is useful, we now check for each parameter whether the default value provided by the ANOVA AMI heuristic, which identifies the best values on average, is also the best or within 0.05 of the best for both AMI and silhouette on *all* datasets. Based on this criterion, we find that for the AMI scran, Seurat, and ZinbWave have one parameter that benefits from dataset-dependent tuning (number of top genes for scran and Seurat; dimension of the latent space for ZinbWave), DCA has two (scale variance and dropout rate), and scVI has three (dimension of the latent space, number of hidden neurons, and number of training epochs), and for the silhouette, scran, Seurat, ZinbWave, and scVI have one parameter that benefits from dataset-dependent tuning (dimension of the latent space for scran and Seurat; number of top genes for ZinbWave; number of training epochs for scVI), and DCA has two (scale variance and dropout rate). Additional file 1: Table S1 and S2 detail the potential gain in dataset-specific tuning for each parameter of each method. This therefore confirms the potential benefit of tuning parameters on each dataset, particularly for autoencoder-based methods.

Tuning parameters in practice

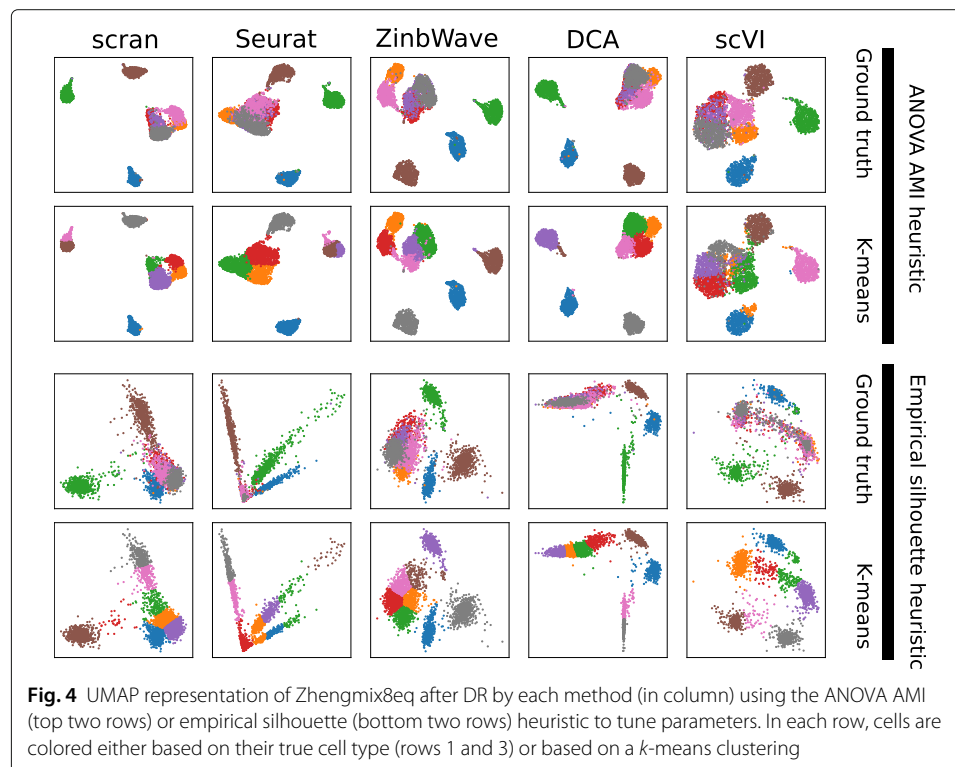
Having shown that DR methods benefit from various degrees of parameter tuning, we now discuss the question of how this can be done in practice. Indeed, our strategy so far to identify the best parameters and evaluate their influence on performance is only possible when one knows the true cell type for each cell in the population, but such an oracle is usually not available in practice. In the absence of such information, one must therefore rely on quantitative heuristics or qualitative validation by domain experts, e.g., by looking at the distribution of cells in the representation space and assessing whether it shows some promising structure such as clusters.

As a first step towards an automated way to tune parameters in a dataset-specific way, we now propose a simple quantitative and objective heuristic to tune parameters, which we call the *silhouette heuristic*, and evaluate its performance on our benchmark. The silhouette heuristic measures how well a distribution of cells in the representation space looks like a possible clustering of distinct cell types. Given a set of cells in a representation space, typically a dataset of cells processed by a DR method with some parameter values, the silhouette heuristic first runs a k -means clustering algorithm on the cells in the representation space, and then computes the silhouette score of the dataset with

respect to the cell types assigned by the k -means clustering. In particular, if the k -means clustering identifies the true cell types, then the silhouette heuristic boils down to the silhouette score with respect to the true cell types. To tune parameters for a DR method on a dataset, we then just compute the silhouette heuristic over a grid of candidate parameter choices and select the values that maximize it. Here, we chose k to be the true number of cell populations, which we already know in advance. In real applications, that number may not be known and has to be estimated with prior knowledge or other heuristics. Thus, our heuristic here only works if the practitioner already knows the true number of populations.

Additional file 1: Figure S5 shows the performance all methods on all datasets when parameters are tuned by maximizing the silhouette heuristic, and Table 2 summarizes the mean performance across datasets. The UMAP projection of the resulting DRs can be seen in Fig. 4.

We can see that the silhouette heuristic works very well for ZinbWave, where it always identifies parameters equal or very close to the best ones in terms of silhouette, and is comparable to the default parameters in terms of AMI. It also works well for all methods for simple datasets like *sc_10x*, *sc_10x_5cl* and *sc_celseq2_5cl*, where it also identifies parameters equal or close to the best ones for the silhouette score. As shown by the good AMI performance, these are cases where the initial k -means clustering recovers the correct clustering with good accuracy. However, there are also cases of surprising failures on easy datasets, for example for DCA on TabulaMuris, where the parameter set selected by the silhouette heuristic has a very bad AMI and silhouette with respect to the true labels, probably because the initial clustering selected by the silhouette heuristic completely fails



to identify the cell types but nevertheless leads to a good empirical silhouette, while simply using the default parameters gives an almost perfect clustering in terms of AMI and a decent silhouette. On the more challenging immune cell datasets, on the other hand, the silhouette heuristic does not seem to be useful (except for ZinbWave). It leads to worse parameters than the default ones for all methods but ZinbWave on the difficult Zhengmix8uneq and Zhengmix5eq datasets, except for scran on the later one. For the easier Zhengmix4eq and Zhengmix4uneq, it leads to better parameters for all methods but scVI. In summary, we see that automatically tuning parameters to try to increase the silhouette using the silhouette heuristic only works well on relatively simple cases, up to possible dramatic errors, but on more challenging situations where there is no clear separation between cell types, then it can lead to disastrous choices by overfitting a bad initial clustering. ZinbWave is an exception where, in our benchmark, the silhouette heuristic gives consistently good results. Proposing other heuristics that really help tune parameters is an important open challenge. In order to stimulate further research in that direction, we provide as supplementary data an original dataset consisting of 5000 embeddings (100 for each single cell dataset and for each method, corresponding to 100 random choices of different parameters, including the default ones and the ones achieving the best AMI and silhouette).

Discussion

In this study, we have systematically compared the performances of five representative and popular DR methods over ten datasets with known experimental ground truth, representing various levels of biological complexity. Importantly, we have extensively investigated how the choice of parameters for these methods influence their performances and discussed various ways to properly tune parameters. This can inform practitioners about both the capacity of these methods, as well as on the amount of work required to properly tune them.

When properly tuned, we did not observe huge differences in performance between the methods, particularly in terms of AMI. Both PCA-based methods (scran and Seurat) are nevertheless outperformed by ZinbWave and both autoencoder-based methods (DCA and scVI), particularly in terms of silhouette. On the other hand, we also found that autoencoder-based methods have more parameters that require careful tuning than ZinbWave and PCA-based methods. We illustrated with the silhouette heuristic that automatic parameter tuning is not always easy when the true cell types are not known and can lead to disastrous results. Interestingly, a similar conclusion was reached by [15], who highlighted the impact of parameters choices in a variational autoencoder-based model, and the need to tune them.

We benchmarked DR methods using downstream analysis-agnostic metrics, mostly for simplicity and because of a lack of ground truth, other than simulations, for tasks such as trajectory inference. It would be interesting to investigate in further studies how well our metrics translate to downstream applications: in particular which metric out of the AMI and silhouette correlates best with downstream performances.

An interesting result of our study is the large drop in performance, for all methods, on the immune cell mix data compared to the cell lines. This shows that good performances in the later does not necessarily translate to good performances in the former. In particular, the performances on Zhengmix8eq and Zhengmix8uneq showed that the methods

failed to properly separate the various T cells populations, probably due to their relative similarity compared to the other cell populations present in the datasets. Being able to separate similar populations is of utmost relevance when investigating, for example, early tumor development, where the tumor cell population still displays a transcriptome very similar to that of cells of the organ of origin. For example in the case of triple negative breast cancer, in which tumor cells originate from normal luminal cell populations, it is crucial to be able to distinguish the various states the tumor cells undergoes towards full transformation, in order to properly target these cells at an early stage of the disease. Our study shows that efforts are still needed to develop methods able to robustly discover cell populations in complex mixtures.

Material and methods

Clustering algorithms

To perform the clustering we used 3 different algorithms: k -means, hierarchical clustering with Ward's linkage, and Louvain. Both k -means and Ward were run with correct number of clusters and default parameters using scikit-learn's implementation [25], version 0.21.3. For Louvain, the nearest neighbors graph construction was done with scanpy (version 1.3.8) using default parameters, and the clustering was also run with default parameters using the "taynaud" flavor. Note that since we had to automatically run Louvain on all embeddings, as done in [23], we could not properly tune the resolution nor the size of the neighborhood and thus Louvain could either overcluster or undercluster.

Performance metrics

Given a set of cells with given ground truth labels, we consider two metrics to measure how well a mapping of those cells in a representation space fits the ground truth labels.

The first metric is the mean *silhouette*, defined as the average over all cells of each cell's silhouette. The silhouette of a given cell x is defined as

$$\text{silhouette}(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))},$$

where $a(x)$ is the average Euclidean distance between x and the other cells of the same class, and $b(x)$ is the average Euclidean distance between x and cells in the closest different class. We used the implementation of scikit-learn.

The second metric is the AMI [26], which measures how well the ground truth clustering U matches the clustering V found by a clustering algorithm (such as k -means) in the representation space. The AMI is formally defined as:

$$\text{AMI}(U, V) = \frac{\text{MI}(U, V) - \mathbb{E}[\text{MI}]}{\text{mean}(H(U), H(V)) - \mathbb{E}[\text{MI}]},$$

where $\text{MI}(U, V)$ is the mutual information between both clustering U and V , H is the entropy function, and $\mathbb{E}[\text{MI}]$ is the expected mutual information between U and a random clustering V . We used scikit-learn's implementation with default parameters for both the k -means algorithm (setting k equal to the ground truth number of classes) and to compute the AMI.

The ARI [27] is another metric that can be used to compare how two clusterings match each other. It is formally defined as:

$$\text{ARI}(U,V) = \frac{\text{RI}(U,V) - \mathbb{E}[\text{RI}]}{\max(\text{RI}) - \mathbb{E}[\text{RI}]},$$

where $\text{RI}(U, V)$ is the Rand index, i.e., the fraction of pairs of samples which are either in the same group or in different groups in both U and V , $\mathbb{E}[\text{RI}]$ is the expected Rand index between U and a random clustering V , and $\max(\text{RI})$ is the largest possible Rand index between U and any V . We also used scikit-learn's implementation of the ARI and compared the clustering obtained with k -means with the ground truth labels as we did for the AMI.

Statistical analysis

To analyze results we used R (version 3.6) to perform T tests and run ANOVA analysis with the `AovSum` function from the `FactoMineR` package (version 2.3) [28]. All parameter values were turned into factors for the ANOVA analysis. To compare the methods presented in Table 2, as shown in Additional file 1: Figure S16–S17, we used the `wilcoxon` function from the `scipy` package [29] (version 1.14), with default parameters, except for alternative which was set to “greater” in order to have a one-way test. Note that we only had 10 samples, which is small for that test.

Computational methods

The five DR methods were downloaded from their canonical package. We used `scrn` version 1.14.6, `Seurat` version 3.1.5, `ZinbWave` version 1.8.0, `DCA` version 0.2.3, and `scVI` version 0.4.0. We followed either the tutorials or vignettes available at the time of download for each methods to use them. The selection of parameters to tune was based on the arguments of the functions called in these tutorials. Methods dependent on a random seed, `DCA` and `scVI`, were run on five seeds, and we averaged their metrics in order to reduce the effect of a single good or bad seed.

UMAP plots

The UMAP plots were generated using `scanpy`'s implementation with default parameters, run on all the dimensions of the low dimension representation.

Supplementary information

Supplementary information accompanies this paper at <https://doi.org/10.1186/s13059-020-02128-7>.

Additional file 1: Contains additional figures and tables for the paper.

Additional file 2: Review history.

Peer review information

Barbara Cheifet was the primary editor on this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Review history

The review history is available as Additional file 2.

Authors' contributions

All authors designed the study, analyzed the results, and wrote the manuscript. FR did all the implementation and experiments. All authors read and approved the final manuscript.

Funding

The authors have no funding to declare.

Availability of data and materials

The four cell lines datasets come from [12] and were downloaded from https://github.com/LuyiTian/sc_mixology in April 2019. The TabulaMuris dataset is an in silico mixture containing all the cells from four tissues sequenced with Smart-Seq2 from the Tabula Muris consortium [22] and was downloaded with the TabulaMurisData Bioconductor package, version 1.2.0. Zhengmix4eq, Zhengmix4uneq, and Zhengmix8eq come from [11] and were downloaded from the DuoClustering2018 Bioconductor package, version 1.2.0. We generated Zhengmix5eq and Zhengmix8uneq following the same procedure in order to have more complex datasets. All datasets were subject to the same quality control pipeline, using scater [16], version 1.14.6. We removed cells three median absolute deviations (MAD) under the mean in counts and expressed genes, as well as those three MAD above the mean in percentage of mitochondrial reads. The code and data used in this manuscript, as well as the 5,000 pre-computed embeddings provided as a benchmark to develop new heuristics for parameter selection, are freely available under an Apache License 2.0 [30, 31].

Ethics approval and consent to participate

The authors declare that no ethics approval was required.

Competing interests

The authors declare that they have no competing interests.

Author details

¹Google Research, Brain team, 75009 Paris, France. ²CNRS UMR3244, Institut Curie, PSL Research University, 75005 Paris, France. ³Translational Research Department, Institut Curie, PSL Research University, 75005 Paris, France.

Received: 1 May 2020 Accepted: 3 August 2020

Published online: 24 August 2020

References

- Kolodziejczyk AA, Kim JK, Svensson V, Marioni JC, Teichmann SA. The technology and biology of single-cell RNA sequencing. *Mol Cell*. 2015;58(4):610–20.
- Zeisel A, Muñoz-Manchado AB, Codeluppi S, Lönnerberg P, La Manno G, Juréus A, Marques S, Munguba H, He L, Betsholtz C, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*. 2015;347(6226):1138–42.
- Tasic B, Menon V, Nguyen TN, Kim TK, Jarsky T, Yao Z, Levi B, Gray LT, Sorensen SA, Dolbeare T, Bertagnolli D, Goldy J, Shapovalova N, Parry S, Lee C, Smith K, Bernard A, Madisen L, Sunkin SM, Hawrylycz M, Koch C, Zeng H. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nat Neurosci*. 2016;19(2):335–46.
- Macosko EZ, Basu A, Satija R, Nemes J, Shekhar K, Goldman M, Tirosh I, Bialas AR, Kamitaki N, Martersteck EM, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*. 2015;161(5):1202–14.
- Schiebinger G, Shu J, Tabaka M, Cleary B, Subramanian V, Solomon A, Gould J, Liu S, Lin S, Berube P, et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*. 2019;176(4):928–43.
- Kharchenko PV, Silberstein L, Scadden DT. Bayesian approach to single-cell differential expression analysis. *Nat Methods*. 2014;11(7):740–742.
- Hicks SC, William Townes F, Teng M, Irizarry RA. Missing data and technical variability in single-cell RNA-sequencing experiments. *Biostatistics*. 2017;19(4):562–78.
- Hwang B, Lee JH, Bang D. Single-cell RNA sequencing technologies and bioinformatics pipelines. *Exp Mol Med*. 2018;50(8):96.
- Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol Syst Biol*. 2019;15(6):e8746.
- Sun S, Zhu J, Ma Y, Zhou X. Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. *Gen Biol*. 2019;20:269.
- Duò A, Robinson MD, Soneson C. A systematic performance evaluation of clustering methods for single-cell RNA-seq data. *F1000Research*. 2018;7:1141.
- Tian L, Dong X, Freytag S, Lê Cao K-A, Su S, JalalAbadi A, Amann-Zalcenstein D, Weber TS, Seidi A, Jabbari JS, et al. Benchmarking single cell rna-sequencing analysis pipelines using mixture control experiments. *Nat Methods*. 2019;16:479–87.
- Vieth B, Parekh S, Ziegenhain C, Enard W, Hellmann I. A systematic evaluation of single cell RNA-seq analysis pipelines. *Nat Commun*. 2019;10:4667.
- Saelens W, Cannoodt R, Todorov H, Saey Y. A comparison of single-cell trajectory inference methods. *Nat Biotechnol*. 2019;37(5):547.
- Hu Q, Greene CS. Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell RNA transcriptomics. *Pac Symp Biocomput*. 2019;24:362–73.
- McCarthy DJ, Campbell KR, Lun ATL, Wills QF. Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. *Bioinformatics*. 2017;33(8):1179–86.
- Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat Biotechnol*. 2018;36(5):411.
- Risso D, Perraudeau F, Gribkova S, Dudoit S, Vert J-P. A general and flexible method for signal extraction from single-cell RNA-seq data. *Nat Commun*. 2018;9(1):284.
- Eraslan G, Simon LM, Mircea M, Mueller NS, Theis FJ. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat Commun*. 2019;10(1):390.

20. Lopez R, Regier J, Cole MB, Jordan MI, Yosef N. Deep generative modeling for single-cell transcriptomics. *Nat Methods*. 2018;15(12):1053.
21. Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, Ziraldo SB, Wheeler TD, McDermott GP, Zhu J, et al. Massively parallel digital transcriptional profiling of single cells. *Nat Commun*. 2017;8:14049.
22. Schaum N, Karkanas J, Neff NF, May AP, Quake SR, Wyss-Coray T, Darmanis S, Batson J, Botvinnik O, Chen MB, et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris: the Tabula Muris Consortium. *Nature*. 2018;562(7727):367.
23. Chen H, Lareau C, Andreani T, Vinyard ME, Garcia SP, Clement K, Andrade-Navarro MA, Buenrostro JD, Pinello L. Assessment of computational methods for the analysis of single-cell ATAC-seq data. *Genome Biol*. 2019;20(1):1–25.
24. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. *J Stat Mech Theory Exp*. 2008;2008(10):P10008.
25. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. *J Mach Learn Res*. 2011;12:2825–30.
26. Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J Mach Learn Res*. 2010;11(Oct):2837–54.
27. Hubert L, Arabie P. Comparing partitions. *J Classif*. 1985;2(1):193–218.
28. Lê S, Josse J, Husson F, et al. Factominer: an R package for multivariate analysis. *J Stat Softw*. 2008;25(1):1–18.
29. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nat Methods*. 2020;1–12.
30. Raimundo F, Vallot C, Vert JP. Code for "Tuning parameters of dimensionality reduction methods for single-cell RNA-seq analysis". Zenodo. 2020. <https://doi.org/10.5281/zenodo.3966952>.
31. Raimundo F, Vallot C, Vert JP. Data for "Tuning parameters of dimensionality reduction methods for single-cell RNA-seq analysis". Zenodo. 2020. <https://doi.org/10.5281/zenodo.3966234>.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions

