Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

The necessity of data-driven optimization of the number of principal components in scRNA-seq analysis - examples on real and simulated data

Manja Grønberg
Technical University of Denmark
Alisa Pavel
University of Southern Denmark
Line Clemmensen
Lkhc@math.ku.dk
University of Copenhagen

Research Article

Keywords:

Posted Date: October 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10587521/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

The necessity of data-driven optimization of the number of principal components in scRNA-seq analysis - examples on real and simulated data.

Manja Grønberg[1], Alisa Pavel[1,2], Line Clemmensen[1,3*]

[1]Dept. of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads B324, Kgs. Lyngby, 2800, Denmark.

[2]Dept. of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, Odense M, 5230, Denmark.

[3]Dept. of Mathematical Sciences, University of Copenhagen, Universitetsparken 5, København Ø, 2100, Denmark.

*Corresponding author(s). E-mail(s): lkhc@math.ku.dk;
Contributing authors: mgegr@dtu.dk; pavel@imada.sdu.dk;

Abstract

Background: Single-cell RNA-sequencing (scRNA-seq) typically monitors thousands of genes, resulting in high-dimensional data. Dimensionality reduction methods are therefore essential, with principal component analysis (PCA) being among the most widespread. PCA reduces dimensionality by selecting the first $\boldsymbol{k}$ principal components to create a lower-dimensional embedding. Common recommendations suggest using 10-50 components, and the canonical pipelines have a default of 50. However, no justification for this choice exist. Other resources suggest data-driven methods to choose the number of components, however these methods have not been systematically evaluated for scRNA-seq analysis.

Results: We therefore investigate the rationale for using a default number of components and evaluate eight data-driven methods for choosing the number of components in scRNA-seq analysis. Our study includes 25 real scRNA-seq datasets and 37 simulated datasets spanning diverse data characteristics. PCA embeddings are assessed by clustering-based downstream analyses (i.e.

tasks based on KNN-graphs), with respect to clustering score, neighborhood preservation, convergence, assumptions, and computational cost.
Conclusion: Evaluating clustering performance across a range of PCA embedding sizes, we find that most datasets have a broad optimal range between 10 and 100 components. However, a few datasets behave differently, suggesting that fixed defaults are insufficient as scRNA-seq data increase in size and complexity and evolve with changing protocols. Although the general recommendation of between 10-50 components, including the default 50, performs well for many datasets, we argue that data-driven selection of the number of components is a necessity for scRNA-seq analysis. Evaluation of the data-driven methods leads to practical recommendations for data-driven methods.

Keywords: benchmarking, scRNA-seq data, dimensionality reduction, PCA, number of components, clustering

1 Background

High-throughput methods, such as single cell RNA sequencing (scRNA-seq), allow unique insights into cell diversity, population heterogeneity and cell behavior. However, the analysis of these data is challenging due to their complexity, sparsity, noise, and not least high dimensionality [1]. The human coding genome is estimated to be around 20000 genes [2], but many more gene products are assumed to play a significant role in biological processes. Therefore, measuring thousands to hundreds of thousands of genes (and their products) is needed to investigate and potentially understand the complexity of biological processes and systems. This results in scRNA-seq datasets of high dimension, both in samples (cells) and features (genes), where it is not uncommon that the number of features is larger than the number of cells, which can pose a challenge for classical machine learning (ML) and data analysis methods. The large feature space, and the amount of data in general, further leads to high computational costs and difficulty in human interpretability of the data.

Dimensionality reduction methods are therefore essential for scRNA-seq data. A popular method to reduce data dimensionality, reduce noise, and lower downstream

computational cost is principal component analysis (PCA) [3]. Although this method dates back to 1901 [4], the authors of [3] found it to be recommendable in most settings compared to a number of more modern dimensionality reduction methods. PCA computes a sequence of orthogonal linear approximations, denoted principal components (PC) which are ordered according to the amount of variance in the original data they account for. The reduction in dimensionality is obtained by choosing the first k components, where k is much lower than the original dimension. We will denote this low-dimensional representation of the data the PCA embedding of size k. Selecting too few components can lead to a loss of local diversity (i.e., cell population diversity), and as a result, vital biological information may not be considered in downstream analysis. In contrast, using too many components increases the computational cost of downstream tasks and risks including technical noise.

In canonical analysis pipelines for scRNA-seq data, such as those provided in Seurat (v5) [5] and Scanpy (v1.11) [6], the PCA embedding is used for a multitude of downstream applications, such as clustering (e.g., Leiden algorithm [7]), visualization (e.g., uniform manifold approximation projection (UMAP) [8]), trajectory analysis (e.g., Monocle [9]), and differential gene expression analysis. The choice of components is therefore essential for the outcome and conclusion of the scRNA-seq analysis. However, choosing the optimal number of components is not easy and we argue that it is both dataset- and task-specific. The selection of components is further complicated by the implementation of downstream tasks in the packages, which back-end runs a PCA with 50 components (default), without taking into account the complexity of the dataset or task. For example, when bulding the k-nearest neighbor (KNN) graph in Seurat or Scanpy, the graph is typically constructed on the default PCA dimensions. Consequently, downstream analyses such as Leiden and Louvain clustering [7, 10] as well as UMAP visualization [8] are performed in this PCA space.

A common recommendation is to choose between 10 and 50 components (see tutorials [11–13]), aligning with the default value in these canonical analysis pipelines and its frequent use in practice [13–15]. This opens an important question: Why do we operate with a default number of components and is this appropriate?

In the first part of this paper we therefore investigate the impact of the size of the PCA embedding on the downstream task clustering (i.e., tasks based on KNN graphs) and the indication of a default size of the PCA embedding. Our investigation cover both real scRNA-seq data from publicly available repositories and simulated scRNA-seq data, for which the ground truth is known and the labels accurately reflect the true classes.

Other resources suggest using data-driven methods to choose the number of components [16, 17]. While several data-driven methods have been suggested, these methods have not been systematically evaluated on scRNA-seq data. To the best of our knowledge only the preprint by [18] focuses on scRNA-seq data and evaluates some of these methods on simulated data.

In the second part of this paper, we therefore evaluate and discuss methods for choosing the number of principal components in scRNA-seq analysis. We compare a range of computational methods which represent different approaches and strategies for estimation. We investigate the eigenvalue-based methods including Adjusted Correlation Thresholding (ACT), two versions of Eigengaps (EG and EG-bound), Explained Variance (EV) and Scree plot (SCREE); the hypothesis-based method Kritchmann-Nadler (KN); simulation-based methods Jackstraw (JS) and Parallel Analysis (PA); as well as the default 50 components (PC50).

To our knowledge, this is one of the first studies extensively exploring the indication of a default size of the PCA embedding for scRNA-seq analysis as well as discussing and evaluating data-driven methods in the context of scRNA-seq data, in order to answer the question of how to choose the number of components.

2 Results

The results are split into two main sections. The first evaluates how different sizes of the PCA embedding affect downstream clustering, while the second section assesses data-driven methods to determine the size of the PCA embedding.

We focus on clustering as the downstream analysis and build the KNN-graph on the reduced PCA space followed by the Leiden algorithm, consistent with the standard pipeline. We measure the clustering quality by the adjusted rand index (ARI) between the Leiden clustering labels and the true labels. Clustering quality is highly sensitive to the number of clusters found by the Leiden algorithm, which is primarily controlled by the resolution parameter [19]. We observe that the number of clusters varies with the size of the PCA embedding when the resolution parameter is fixed, making joint optimization of the resolution and the PCA dimension necessary. Since our purpose of clustering is to evaluate the quality of the PCA embedding, we present results using resolutions aligned with the true number of clusters. Although this approach is not practically feasible, as it relies on extrinsic information, results based on a practical alternative, namely the silhouette score, are provided in the appendix. A more elaborate discussion is given in section 4.5.

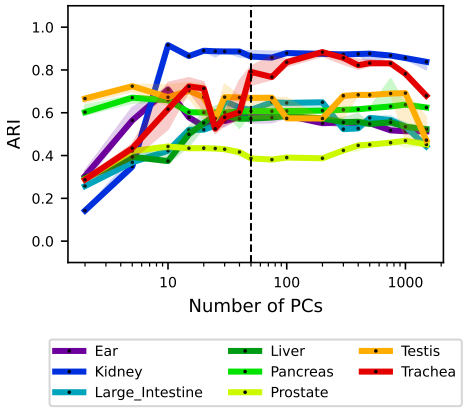
We use 25 real scRNA-seq datasets along with 37 simulated datasets for evaluation. The real datasets come from different labs, are obtained under different conditions, investigate different research questions, and thus cover a wide variety of scRNA-seq data while the simulated data are obtained by four different simulators (see section 4.2 for details).

2.1 The impact of the size of the PCA embedding on clustering performance

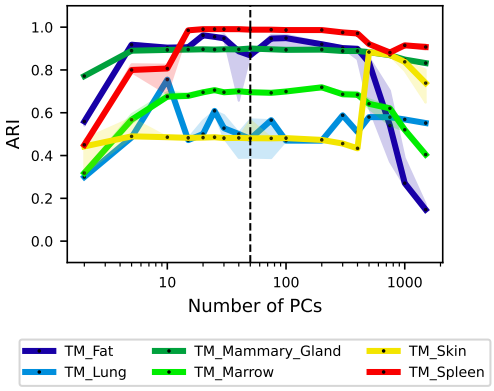
Fig. 1 presents the impact of the size of the PCA embedding on clustering performance (ARI score) on the real scRNA-seq data where colors indicate the individual datasets. For most datasets, the performance profile initially increases, then flattens, and some profiles then decrease at higher dimensions. This behavior is expected according to theory; too few components fail to capture all the information in the data, whereas too many components include excessive amounts of noise. Most notably, most of the performance profiles exhibit a broad range of embedding sizes that yield comparable performances. For the majority, this range lies between 10 and 100 components. The default 50 components conveniently falls within this range, which might be the reason for its widespread adoption. However, while the exact choice of components might not be critical for most datasets, it is critical to choose a value within this optimal range. This range could change as datasets grow in size, quality, and complexity (characteristics) or as laboratory techniques change. Supporting this, Seurat has historically changed its default number of components from 20 components in v2[1] to 50 components in the later versions.

Even though most datasets share these characteristics, there are a few datasets where the optimal range is more narrow (sharper peak) or shifted. For example, the Koh and Kumar datasets in fig. 1c, do not include 50 components in the optimal range. Thus, for some datasets the clustering performance is more sensitive to the size of the PCA embedding and it is important to be careful. Combined, this calls for data-driven methods to determine an appropriate size of the PCA embedding.

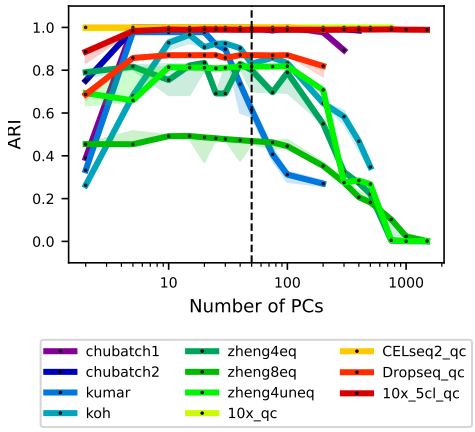
[1] Source code https://cran.r-project.org/src/contrib/Archive/Seurat/.



(a) Tabula Sapiens



(b) Tabula Muris



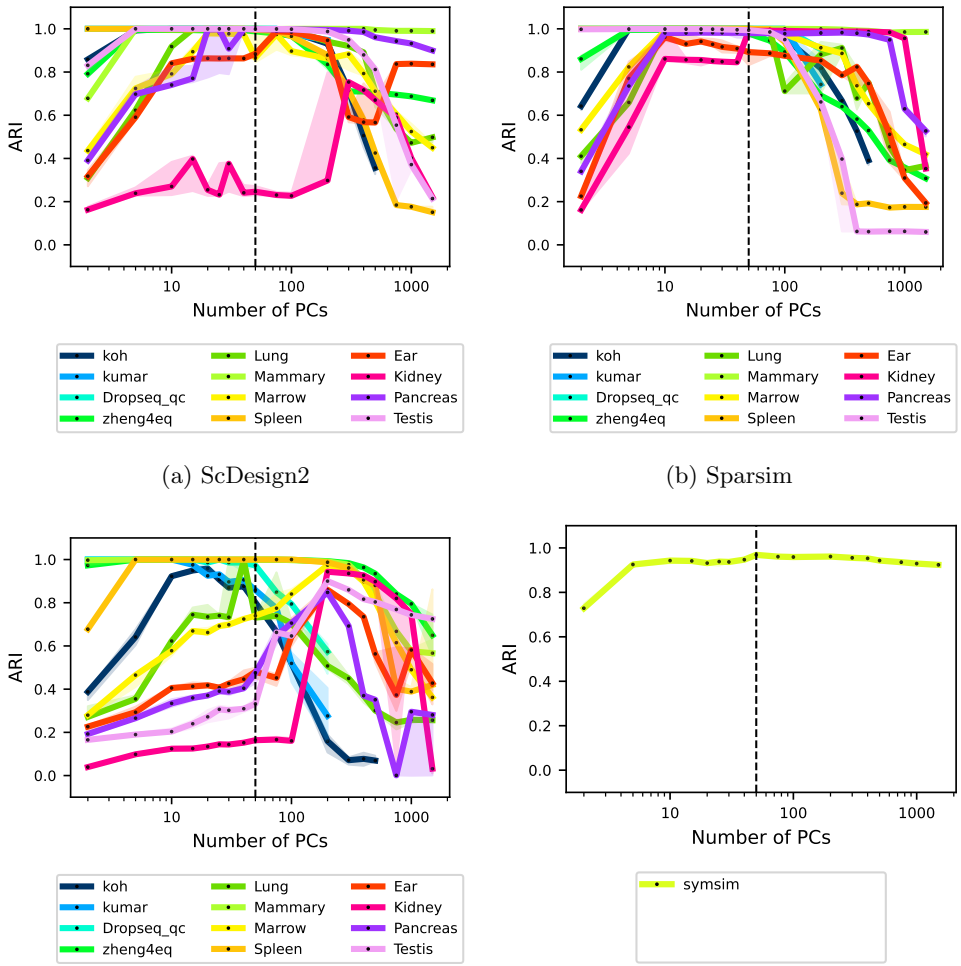
(c) Evaluation

Fig. 1: ARI scores over a range of sizes of the PCA embedding for the real scRNA-seq datasets. Higher ARI scores indicate better clustering performance. The shaded areas represent the 80th percentile across 10 repetitions, and the dotted line indicates the default of 50 PCs.

The results for the simulated datasets in fig. 2 show similar but more pronounced patterns. Note for example that the optimal range for several of the Splatter datasets is narrow and do not include 50 components. For the simulated data, we were able

to artificially construct balanced groups. Under balanced conditions, the performance curves tend to have a wider optimal range including lower dimensionalities (see fig. D8 in the appendix).

Without standardization prior to PCA the optimal range is wider and extends to higher dimensionalities for several real datasets (see eg. Fat, Marrow, and Zheng in figs. D4 and D7 in appendix). Increasing the number of highly variable genes yields similar results to those presented here (see figs. D3 and D6 in appendix).



(a) ScDesign2

(b) Sparsim

(c) Splatter

(d) Symsim

Fig. 2: ARI scores over a range of sizes of the PCA embedding for the simulated scRNA-seq datasets. Higher ARI scores indicate better clustering performance. The shaded areas represent the 80th percentile across 10 repetitions, and the dotted line indicates the default of 50 PCs.

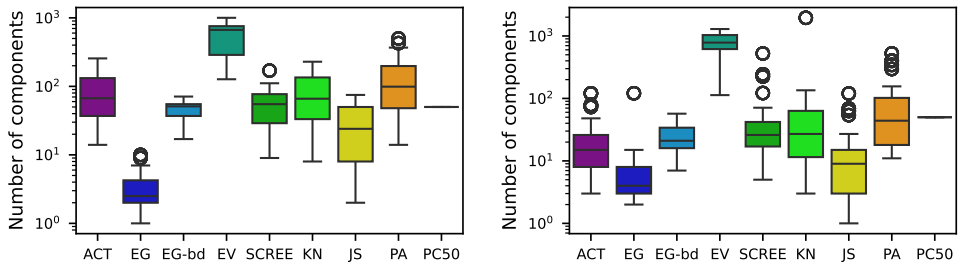
2.2 Computational methods to estimate the size of the PCA embedding

We investigate a wide variety of data-driven methods which have different methodological approaches, different assumptions, and varying complexity to address the problem of finding an appropriate size of the PCA embedding. For comparison, we also include choosing the default 50 components (PC50). A description and discussion of these methods are found in section 4.1.

2.2.1 The number of components

Figure 3 shows that the methods generally do not agree on the optimal number of components for the PCA embedding. In fact, there is a large span between the smallest and largest number of components suggested by these methods even within the same dataset. EG typically chooses fewer components than the other methods (particularly for the real scRNA-seq datasets), followed by JS. The methods ACT, PA, KN, SCREE, EG-bound, and PC50 tend to select a similar range of components, whereas EV generally prefers more components than the other methods. The variation in the number of components selected by EG-bound tends to be small compared to the other methods. This is a consequence of the method's design, which limits the range of components that can be selected.

When the data is not standardized prior to PCA, PA tends to suggest fewer components, but still more components than EG (see fig. D9 in appendix). This may indicate the presence of a shadowing effect in PA (described by [20]), where one factor dominates, causing factors to be excluded as noise when then data is not standardized. For some methods, we observe a slight increase in the optimal number of components when the number of highly variable genes is increased to 5000 (see fig. D10 in appendix).

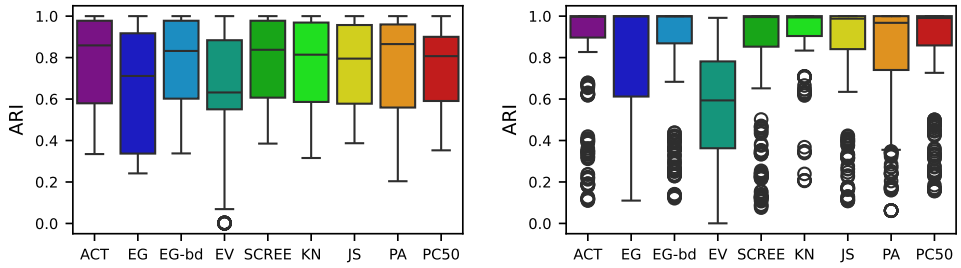


(a) Real scRNA-seq data. (b) Simulated scRNA-seq data.

Fig. 3: Boxplots (logscale) of the optimal number of components computed by each method on the (a) real scRNA-seq datasets and (b) the simulated scRNA-seq datasets.

2.2.2 Clustering performance

Although the methods choose a wide range of components, this variation does not necessarily translate into a large difference in clustering performance (see fig. 4). This is expected based on the wide range of embedding sizes that gave similar clustering performances observed in section 2.1.



(a) Real scRNA-seq data. (b) Simulated scRNA-seq data.

Fig. 4: Boxplots of clustering performance (ARI) of the data-driven methods for (a) all real scRNA-seq data (b) all simulated data sets.

EV performs poorly on most simulated datasets, suggesting that it selects too many components and thereby introduces excess noise into the embedding. However, EV outperforms the other methods for the simulated Splatter datasets Ear, Kidney

and Testis, indicating that some datasets require a large number of components to retain relevant information. When groups are balanced EV never achieves a better clustering score than the other methods (see fig. D15). For several real scRNA-seq datasets including Koh, Kumar, Zheng, Fat, and Marrow, EV also performs worse than the other methods. However, for the Skin data and to some degree the Prostate data, EV is the only method that chooses enough components to retain the information necessary for optimal clustering. Overall, although EV still underperforms relative to the other methods for the real datasets, selecting many components appears less detrimental than for the simulated data.

EG selects fewer components than the other methods, which negatively affects clustering performance, particularly for the real scRNA-seq datasets, possibly reflecting their more complex nature.

ACT, EG-bound, JS, KN, SCREE, PA, and PC50 generally show similar clustering performance across datasets with few exceptions. For example, PA shows reduced clustering performance for the Zheng datasets and the Testis data. No single method selects a superior embedding in terms of clustering score across datasets. KN does not converge (empty cells) for the majority of the simulated datasets and it is therefore difficult to assess the performance of KN on the simulated data.

The patterns are similar when increasing the number of highly variable genes to 5000 or using non-standardized data (see section D in appendix).

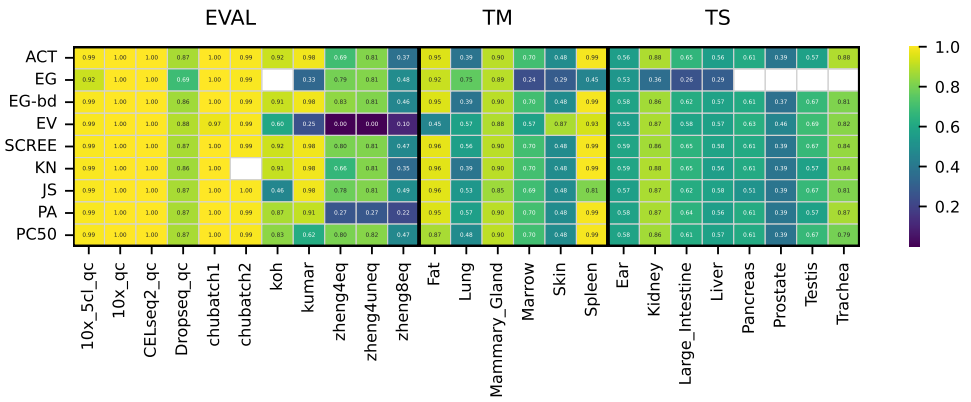


Fig. 5: Heatmap of the ARI scores across the data-driven methods for each real scRNA-seq dataset. Larger values and lighter color indicate better clustering performance. Empty cells indicate that the method did not converge.

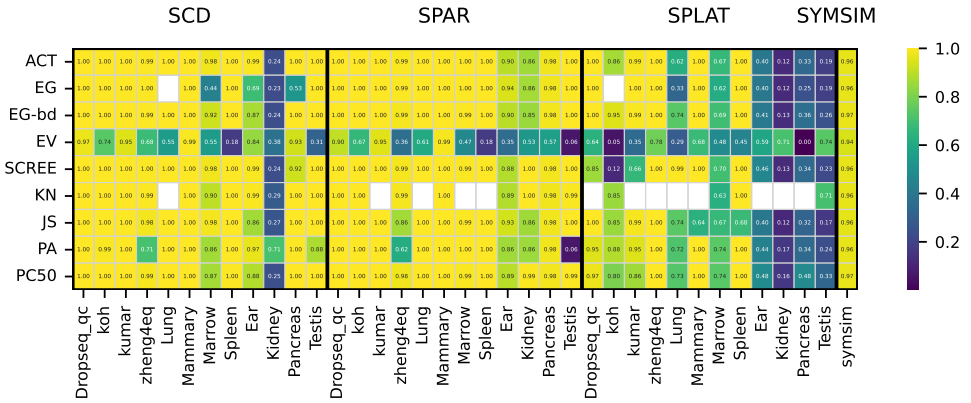


Fig. 6: Heatmap of the ARI scores across the data-driven methods for each simulated scRNA-seq dataset. Larger values and lighter color indicate better clustering performance. Empty cells indicate that the method did not converge.

2.2.3 Neighborhood preserving

Ground truth data are available for the scRNA-seq simulators Sparsim, Splatter, and Symsim. This enables investigation of whether the PCA embeddings preserve local

neighborhoods. We measure this by the trustworthiness score, which compares the k-nearest neighbors in the ground truth data to the k-nearest neighbors in the PCA embedding for each observation. The results for $k = 15$ neighbors are illustrated in fig. 7.

Across nearly all datasets, neighborhoods are best preserved with few components, as EG almost consistently achieves the highest trustworthiness score. ACT, EG-bound, JS, PA, PC50, and SCREE generally give very similar trustworthiness scores, with the exception that PA, PC50, and SCREE perform slightly worse for the Splatter datasets. The embeddings produced by EV do not preserve neighborhoods as well as the other methods. KN fails to converge for many of the datasets, limiting assessment. The results for $k = 50$ neighbors, increased highly variable genes, and non-standardized data prior to PCA are found in section D.4 in appendix. The patterns in these plots are similar.

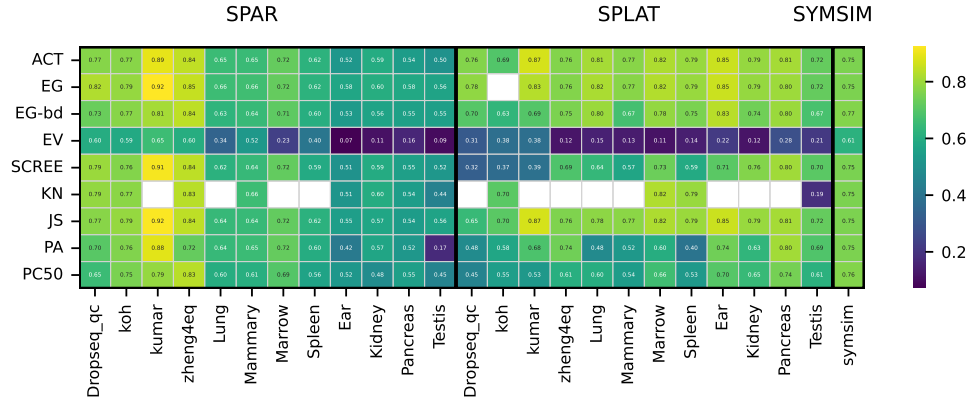
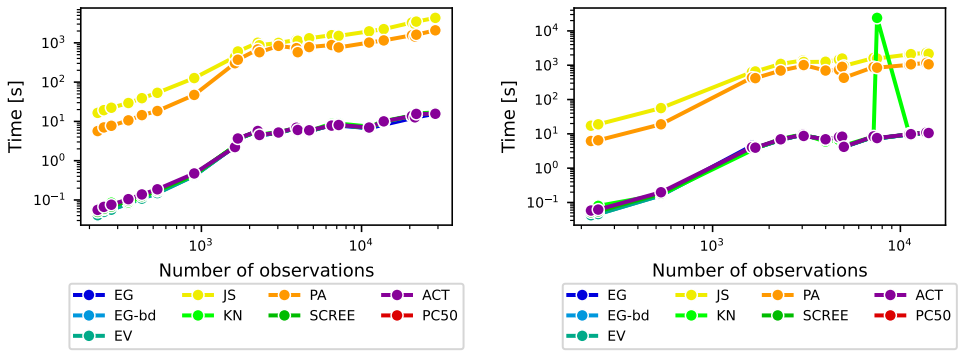


Fig. 7: Heatmap of the trustworthiness using 15 neighbors on each simulated scRNA-seq data set where there exists a ground truth for each method. Higher values and lighter colors indicate better neighborhood preservation. Empty cells indicate that the method did not converge.

2.2.4 Wall-clock time

With increasing sizes of scRNA-seq data, particularly increasing number of cells, (the gene space is typically reduced beforehand by selecting the most highly variable genes, which is also the case here), it is important that the data-driven methods are applicable in terms of computation time. Figure 8 shows the median computation time and scalability (slope) for each method. ACT, EG, EG-bound, EV, and SCREE mainly spend computation time on computing the eigenvalues. Thus, their computation time is very similar. KN is slightly slower, as it requires solving a system of nonlinear equations that grows in size with the number of components. KN can "explode" in computation time, depending on how many iterations it takes to solve the nonlinear system of equations. The computations of PA and JS are slower than the remaining methods, since they depend on simulations which require repetitive computations of the eigenvalues. The scalability (slope in the plots) of the methods is very similar.



(a) Real scRNA-seq data.

(b) Simulated scRNA-seq data

Fig. 8: Log-log plots of the median computation times over the 10 repetitions for each method.

2.3 Practical summary

To summarize the results, we provide a practical overview of the methods and their capabilities in table 1. The methods are divided into categories based on performance for each evaluation criteria. Details of the categorization in tier 1-3 are given in section 4.4. Methods in tier 1 are ranked on top, whereas methods in tier 3 are at the bottom.

EG and EV perform worse than the other methods in terms of clustering. EG improves on the simulated data, possibly owing to the simulated datasets less complex nature and thereby suitability to lower-dimensional embeddings. A drawback of EG is that it fails to converge for some datasets.

In general, clustering performance on the real scRNA-seq datasets is modest, whereas the clustering performance for the simulated datasets is generally better. No single method is consistently better than other methods.

ACT, EG-bound, SCREE, KN, and PC50 achieve the best clustering performances, closely followed by JS and PA. This is the trend across the different pre-processing settings (see tables D6 and D7 in appendix).

KN is placed in tier 3 for convergence success rate, reflecting convergence issues for several datasets. KN also relies on strong distributional assumptions, limiting its practical applicability.

EG-bound and especially PC50 are constrained by fixed limits on the size of the PCA embedding. This reduces flexibility and may negatively affect analysis of scRNA-seq data with characteristics different from those studied here (historical data), making these methods less recommendable in practice, despite their performance across several evaluation criteria.

Overall, ACT, SCREE, and the simulation-based methods JS and PA are recommended. Although the simulation-based methods are computationally slower due to

repeated computations, they scale similarly to the other methods. Since this is typically a one-time computation, the increased runtime may not be a major concern. Note that ACT and JS assume standardized data, so their use depends on whether standardization is appropriate for the analysis (see section 4.3 for a discussion hereof).

Table 1: Summary of the computational methods to estimate the number of components of PCA. These results are based 2000 highly variable genes and standardization prior to PCA. Y=Yes.

		ACT	EG	EG-bd	EV	JS	KN	PA	PC50	SCREE
Real	Clustering accuracy	Tier 2	Tier 3	Tier 2	Tier 3	Tier 3	Tier 2	Tier 2	Tier 2	Tier 2
	Clustering stability	Tier 2	Tier 3	Tier 2	Tier 3	Tier 2	Tier 2	Tier 3	Tier 2	Tier 2
	Number of components	Tier 2	Tier 1	Tier 2	Tier 3	Tier 1	Tier 2	Tier 2	Tier 2	Tier 2
Simulated	Clustering accuracy	Tier 1	Tier 1	Tier 1	Tier 3	Tier 1	Not assessable	Tier 1	Tier 1	Tier 1
	Clustering stability	Tier 1	Tier 2	Tier 1	Tier 3	Tier 1	Not assessable	Tier 2	Tier 1	Tier 2
	Number of components	Tier 1	Tier 1	Tier 1	Tier 3	Tier 1	Not assessable	Tier 2	Tier 2	Tier 1
	Neigh. preserving (k=15)	Tier 1	Tier 1	Tier 1	Tier 3	Tier 1	Not assessable	Tier 2	Tier 2	Tier 2
	Neigh. preserving (k=50)	Tier 1	Tier 1	Tier 1	Tier 3	Tier 1	Not assessable	Tier 2	Tier 2	Tier 2
	Convergence success rate	Tier 1	Tier 3	Tier 1	Tier 1	Tier 1	Tier 3	Tier 1	Tier 1	Tier 1
	Scalability	Tier 1	Tier 1	Tier 1	Tier 1	Tier 1	Tier 1	Tier 1	Tier 1	Tier 1
	Speed	Tier 1	Tier 1	Tier 1	Tier 1	Tier 3	Tier 2	Tier 3	Tier 1	Tier 1
	Distributional assump.						Y			
	Require scaled data	Y				(Y)				
	Dimension constraints			Y					Y	

Tier 1 Tier 2 Tier 3 Not assessable

3 Discussion

We have presented, discussed, and evaluated a range of data-driven methods for selecting the number of principal components in the canonical scRNA-seq analysis pipelines. No single method performs optimally across all settings. A central challenge is to avoid the selection of too few or too many components. Methods such as eigengaps (EG)

and explained variance (EV) illustrate this trade-off. For most datasets, there exists a broad range of embedding sizes, typically between 10 and 100 components, which gain similar clustering performance and thereby seem to retain the same information. This explains why many methods yield comparable clustering performance. However, we find that some methods such as adjusted correlation thresholding (ACT), the scree plot (SCREE), Jackstraw (JS), and parallel analysis (PA) has better properties.

Our evaluation focuses on the canonical pipelines implemented in Scanpy and Seurat. Specifically, we use typical pre-processing steps (log-normalization and selection of highly variable genes) and assess downstream clustering performance using Leiden clustering. Consequently, our results and recommendations are biased towards this choice. We believe this to be a reasonable focus, as these pipelines are widely used in practice. We explore variations in pre-processing, but these do not seem to affect our conclusions drastically.

Methods differ in aspects such as convergence, computation time, assumptions on data, and dimensionality restrictions. Although dimensionality restricted methods (e.g., selecting the default 50 components) perform comparably to non-restricted methods on the datasets considered here, we argue against their use. The characteristics of scRNA-seq data may evolve over time, shifting the appropriate number of (range of) components. Indeed, we already see differences in one or two datasets analyzed here. Relying on dimensionality restricted methods risks overlooking these shifts, potentially leading to suboptimal embeddings and impaired downstream analysis.

4 Methods

Section 4.1 describes methods for choosing the number of components in PCA, while section 4.2 presents the datasets we use for our investigations. Section 4.3 and 4.4 describe our experiments and analysis, while section 4.5 describes different methods to choose the resolution parameter of the Leiden algorithm and a discussion hereof.

4.1 Methods to choose the Number of Components

We will start this section by an introduction to Principal Component analysis (PCA) in order to ensure a consistent mathematical notation. PCA has been described extensively in the literature and we refer to [21] for further reading. PCA is an essential tool in scRNA-seq analysis to reduce the dimensionality of the data. The principal components (PC) of a dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$ are a sequence of linear approximations that are mutually independent (orthogonal) and ordered according to the amount of variance they explain in the original data. The principal components can be found by singular value decomposition of the data matrix such that

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{n \times q}$ and $\mathbf{V} \in \mathbb{R}^{q \times p}$ are orthogonal matrices, and $\mathbf{D} \in \mathbb{R}^{q \times q}$ is a diagonal matrix with the singular values $d_1 \geq d_2 \geq ... \geq d_q$ as diagonal elements. The value q is given by $q = \min(n, p)$. The principal components are then the columns of the matrix $\mathbf{UD}$, while the explained variance of component k is computed from the singular values such that

$$\mathrm{EV}(k) = \lambda_k / \sum_{j=1}^{q} \lambda_j,$$

where

$$\lambda_k = d_k^2 / n, \quad k = 1, ..., q,$$

and λ_k are the eigenvalues.

Alternatively, the principal components can be found by solving the eigenvalue problem for the covariance matrix of $\mathbf{X}$. In the following, we will denote the optimal number of components by K.

Several data-driven methods have been proposed in the literature to estimate the number of principal components to retain in the PCA embedding. This paper considers

8 different methods. We aim at including methods used in the analysis of scRNA-seq data, while also including canonical methods from data science applicable to this type of data. The selected methods represent different methodological approaches to address this problem.

4.1.1 Choose 50 components

One approach, as discussed in the introduction, is to choose the default number of components as implemented in the software packages, namely 50 components. We denote this method PC50. To the best knowledge of the authors, this number (and/or method) has not been validated in the literature, but seems to be a general consensus in the field.

4.1.2 Threshold based methods

The most common approach to estimate the optimal number of components in scRNA-seq data analysis is the scree (SCREE) or elbow plot [17]. This approach dates back to [22]. Both Scanpy v1.11 (scanpy.pl.pca_variance_ratio) and Seurat v5 (ElbowPlot) support functions to construct this plot. The scree plot is a plot of the eigenvalues λ_k associated to the principal components. The method determines the number of components K by locating the point where the curve breaks, also known as the "elbow" point. This point should separate the noise components from the signal components. Investigators usually find this point by visual inspection [4], but there exist several methods which aim to locate this point automatically. Six different methods to estimate the elbow point are implemented in the R-package findPC [23]. They find that the method, which determines the elbow point by maximizing the length of the line perpendicular to the straight line spanned from the first to the last point in the curve,

agrees best with visual inspection. The number of components is then determined by

$$\hat{K} = \arg\max_k \{d_k\},$$

where d_k is the length of the perpendicular line from the point (k, λ_k) to the line spanned between the first point $(1, \lambda_1)$ and the last point (q, λ_q). We implement this method in Python by converting their R-code. Note that these methods always find a bend even if there is no clear bend.

Eigengaps is another method to estimate the optimal number of principal components K. The method was introduced by [24]. Similar to the scree plot, this method looks at the eigenvalues λ_k associated with each component. Eigengaps are the differences (gaps) between eigenvalues. The method sets a threshold τ_{EG} for the eigengaps which separates noise components from signal components and thereby determines the number of components K by

$$\hat{K} = \arg\max_k \{\lambda_k - \lambda_{k+1} \geq \ \tau_{\text{EG}}\}.$$

There are several methods to determine an appropriate value of τ_{EG}. The authors of [24] provide a data-driven iterative method to estimate a threshold based on the shape of the edge of the empirical eigenvalue distribution. This method assumes that the noise is a linear transformation of an independent and identically distributed random variable, but allows for correlation between both observations and features. Additionally it is assumed that $n/p \to c$, i.e. that the observation space grows with the same rate as the feature space. Both of these assumptions might not hold for scRNA-seq data. We implement this method to estimate τ_{EG} in Python and denote

this method EG. The algorithm sets a hyperparameter $k_{\max} < K$ which reduces the search space to $k = 1, ..., k_{\max}$. We choose $k_{\max} = \text{round}(q - 0.1q)$. This method might be sensitive to the choice of $k_{\max}$.

The authors of [25] uses eigengaps for scRNA-seq data. They propose a heuristic to set the threshold τ_{EG} based on the assumption that $K << 100$ and uses the last 20 eigengaps $\lambda_{80} - \lambda_{79}, ..., \lambda_{100} - \lambda_{99}$ to estimate a mean μ and a standard deviation σ of a "typical spacing" between noise eigengaps. They then set $\tau_{\text{EG}} = \mu + 6\sigma$. The authors of [25] mention that this method is not validated theoretically. However, they claim that it works empirically. Note that the choice of τ_{EG} is designed to give an estimate of K which most often lies between 40 and 90. We denote the eigengap method with this choice of τ_{EG} by EG-bound.

Explained variance (EV) also focuses on the eigenvalues and use these to determine the fraction of variance in the original data which is explained by including k components. The method estimates K by setting a threshold τ_{EV} which decides the amount of explained variance to retain such that

$$\hat{K} = \arg\min_k \left\{ \left(\sum_{j=1}^{k} \text{EV}(j) \right) \geq \tau_{\text{EV}} \right\}.$$

This threshold is usually set by the investigator and is often between 70% and 90% [4, p. 113]. The R bioconductor tutorial [26], uses an estimate of the technical variance obtained by the function scran:trendVar to set a threshold for how much of the explained variance to retain. The paper which introduces this method to estimate technical variance [27] is not peer-reviewed and the method to estimate τ_{EV} is only published as a tutorial. As there does not seem to be rigorous evidence of this approach, we will reside to the common practice in data science and choose a

threshold. Here we choose $\tau_{\text{EV}} = 0.8$.

Correlation based methods are commonly applied in data science. These methods assume the data to be standardized such that each feature has mean 0 and standard deviation 1 (solving the eigenvalue problem for the covariance matrix of the standardized data is similar to solving the eigenvalue problem for the correlation matrix of the non-standardized data). In contrast to covariance based methods, these methods are scale-invariant. Kaiser's criterion uses the eigenvalues of the correlation matrix to estimate the number of PCs as

$$\hat{K} = \arg\max_k \{l_k > 1\},$$

where l_k is the kth eigenvalue of the correlation matrix. This method is known to overestimate K [28]. The authors of [29] argue that this method cannot be consistent in high-dimensional settings as the sample correlation matrix is inconsistent.

We implement Adjusted Correlation Thresholding (ACT) [29] which develops a method that is consistent in high-dimensional settings, but still has the simple form with no parameter tuning similar to the Kaiser's criterion. The method bias-corrects the estimated eigenvalues and adjust the threshold to find K by

$$\hat{K} = \arg\max_k \left\{l_k^C > s\right\}, \tag{1}$$

where $s = 1 + \sqrt{p/(n-1)}$ and l_k^C is the bias corrected eigenvalues. The bias corrected eigenvalues are found by

$$l_k^C = -\frac{1}{\underline{m}_{n,k}(l_k)},$$

where

$$\underline{m}_{n,k}(z) = -(1-\rho_{k,n-1})z^{-1} + \rho_{k,n-1}m_{n,k}(z),$$

and

$$m_{n,k}(z) = (p-k)^{-1}\left[\sum_{j=k+1}^{q}(l_j - z) + ((3l_k + l_{k+1})/4 - z)^{-1}\right],$$

$$\rho_{k,n-1} = (p-k)/(n-1).$$

We convert the R-code provided by the authors to Python. Similar to the code implemented in R, we restrict the search space for k in equation (1) to $k = 1, ..., k_{\max}$. We choose $k_{\max} = \text{round}(q - 0.1q)$.

4.1.3 Hypothesis based methods

A second category of methods includes methods which are based on hypothesis testing. These methods often pose rigorous distributional assumptions on the data, which might not be valid for scRNA-seq data. To the best of our knowledge, these methods are not widely used in analysis of scRNA-seq data, maybe due to the rigorous assumptions or the lack of availability. Kritchman-Nadler (KN) presented a method to estimate K which uses a sequence of nested hypothesis tests [30] to separate noise components from signal components, such that

$$\begin{aligned} \text{H}_0: &\quad \lambda_{k+1} \text{ is the largest noise eigenvalue,} \\ \text{H}_A: &\quad \lambda_k \text{ is the largest noise eigenvalue,} \end{aligned}$$

for $k = 1, ..., q-1$. The method assumes uncorrelated homoscedastic Gaussian noise with mean zero and variance σ^2. This is generally not true for the raw-counts, but the standard log-transformation during pre-processing of the scRNA-seq data attempts

to achieve this. The KN method addresses the problem that when data are high-dimensional, i.e. when the ratio p/n is large, then the spread of the noise eigenvalues can be quite large.

The nested hypothesis tests are based on a result from [31] showing that in the limit when $n, p \to \infty$ and the ratio p/n is constant then the distribution of the largest noise eigenvalue, denoted λ_*, converges to a Tracy-Widom distribution such that

$$P\left(\lambda_* < \sigma^2\left(\mu_{n,p} + s\sigma_{n,p}\right)\right) \sim F_1(s)$$

where F_1 is the Tracy-Widom distribution of order 1. Letting

$$\mu_{n,p} = \frac{1}{n}\left(\sqrt{n-1/2} + \sqrt{p-1/2}\right)^2,$$
$$\sigma_{n,p} = \frac{1}{n}\left(\sqrt{n-1/2} + \sqrt{p-1/2}\right)\left(1/(\sqrt{n-1/2}) + 1/(\sqrt{p-1/2})\right)^{\frac{1}{3}}$$

gives a convergence rate of $\mathcal{O}_p(p^{-2/3})$. It has been shown both numerically and theoretically that this result is a good approximation when p and n are large [30].

As the number of signal components K is unknown, we cannot estimate the noise variance unbiased, but have to approximate the noise variance σ^2 for each k by $\sigma^2_{KN}(k)$. This quantity is obtained by solving the system of $k+1$ non-linear equations in $k+1$ unknowns $\rho_1, ..., \rho_k$, and $\sigma^2_{KN}(k)$, such that

$$\sigma^2_{KN}(k) - \frac{1}{p-k}\left[\sum_{j=k+1}^{q}\lambda_j + \sum_{j=1}^{k}(\lambda_j - \rho_j)\right] = 0,$$
$$\rho_j^2 - \rho_j\left(\lambda_j + \sigma^2_{KN}(k) - \sigma^2_{KN}(k)\frac{p-k}{n}\right) + \lambda_j\sigma^2_{KN}(k) = 0,$$

where $\sigma^2_{KN}(k)$ is constrained to be positive. Thus, K is estimated as

$$\hat{K} = \arg\min_k \left\{ \lambda_k \leq \sigma^2_{KN}(k) \left(\mu_{n,p-k} + s(\alpha)\sigma_{n,p-k} \right) \right\} - 1,$$

where α is the significance level and $s(\alpha)$ is the $(1-\alpha)$'th percentile of the Tracy-Widom distribution. We use a significance level of $\alpha = 0.5\%$ consistent with the original paper. For further information see [30].

4.1.4 Simulation based methods

Simulation based methods represent another approach to estimate the number of components. These methods are usually computationally expensive as they require several simulations to get reliable results, but often relies on fewer assumptions on the data, making them suitable for scRNA-seq data. In data science, one of the canonical methods is Parallel Analysis (PA) which was first described by [32]. The most commonly used version of this method is described by [33] (see also [21, p. 538]). This method creates a number of independent datasets with the same variance as the original dataset by permuting the columns of the original dataset and uses these to estimate a threshold which separates noise components from signal components. The method successively compares the kth eigenvalue of the original dataset to the ϕth percentile of the kth eigenvalues of the permuted data. If the k'th eigenvalue of the original dataset is larger than the ϕth percentile, $q_\phi()$, of the kth eigenvalues of the permuted data, then the kth factor is kept. That is

$$\hat{K} = \arg\max_k \{ \lambda_k \geq q_\phi(\tilde{\lambda}_{r,k}) \},$$

where $\tilde{\lambda}_{r,k}$ is the kth eigenvalues of the r permuted datasets. We use $r = 100$ and $\phi = 0.9$.

The most obvious disadvantage of this method is that it is computationally expensive as it requires computation of eigenvalues for several potentially large matrices. The authors of [20] additionally describe a shadowing effect of this method, which means that one large factor may cause preceding factors to be excluded as noise.

A simulation based method which is commonly cited and proposed (see eg. [16, 34, 35]) for analysis of scRNA-seq data is Jackstraw [36]. The Jackstraw procedure was originally presented by [37] as a method to determine if a specific variable is significantly associated to a set (subset) of the principal component. The method thus produces a vector of p-values, testing whether there is correlation between the jth variable and any component for $j = 1, ..., p$. The method thus make one test for each variable. The original implementation of this method uses PA to estimate the number of principal components and then applies the Jackstraw procedure to determine if the variables are significantly associated with these components.

The paper [36] argues that a principal component should only be included if it is significantly associated with a gene (a variable in the original data set). To this end, the authors of [36] adjust the original version of Jackstraw. Opposite of the original version, this version constructs tests, testing whether there is an association between each principal component and any gene (variable), resulting in a test for each principal component. This version of jackstraw is implemented in Seurat.

To the best knowledge of the authors, the adjusted Jackstraw method implemented in Seurat, has not been documented neither statistically nor technically in a published paper. Only the two tutorials documenting the use of the jackstraw method in Seurat (link to seurat tutorial, bioconductor tutorial) are available. We present the Jackstraw algorithm implemented in Seurat in algorithm 1 in appendix and refer to the original paper [37] for the theoretical foundation. The functions in Seurat are *Jackstraw()* and

ScoreJackstraw(). We have implemented the Seurat code in Python by converting the backend R-code from Seurat to Python.

The source code for both the original paper (see the jackstraw package v 1.3.21 on cran) and the Seurat version (see the Seurat package v. 5.5.0 on cran) assume standardized data such that each feature has mean 0 and standard deviation 1 (solving the eigenvalue problem for the covariance matrix of the standardized data is similar to solving the eigenvalue problem for the correlation matrix of the non-standard data). Although there seem to be no formal requirements for standardized data in the publications, we will only apply this method to standardized data. Note that, according to the newest Seurat vignette (v5) [38] this method is no longer routinely used in scRNA-seq analysis.

4.2 Datasets

The goal of the data collection has been to cover a wide variety of scRNA-seq data from different labs, under different conditions and investigating different research questions. We consider 25 real scRNA-seq datasets in total from different sources: 6 Tabula Muris (TM) v2 [39] datasets, 8 Tabula Sapiens (TS) v2 [40] datasets, and 11 datasets from the papers [3] and [41] (EVAL), which evaluates dimensionality reduction and clustering methods for scRNA-seq data. In order to ensure that the cell labels are not biased towards previous analysis (a specific analysis method/pipeline), we only include datasets where the corresponding publication states that these labels are not determined data-driven. This requirement is in line with prior evaluation studies on clustering [3, 41]. We did not include datasets with more than 40000 observations, as they were not computationally feasible for these experiments. This should be within the common size of scRNA-seq datasets according to [42]. Data and their characteristics are described in detail in appendix A.

In addition to the real scRNA-seq data we also generate simulated data from four different published scRNA-seq data simulation methods. All of them were ranked among the top simulation methods in [43], which benchmarked a wide range of published scRNA-seq data simulation methods. In addition, it has been a requirement that the simulation methods could accommodate multiple cell groups. Where possible, we estimated the parameters for the simulation based on real scRNA-seq datasets. This should ensure a wide variety both in terms of simulation models and parameters. In total, we simulate 37 datasets: 12 datasets from each of the R-packages *Sparsim* [44] (SPAR), *ScDesign2* [45] (SCD), and *Splatter* [46] (SPLAT), and 1 dataset from the R-package *Symsim* [47] (SYMSIM). We consider simulated scRNA-seq data in order to have ground truth data, where we know that the labels reflect true classes in data. A challenge of simulated data is that it may not accurately reflect the complexity and characteristics of real scRNA-seq data [1] and these datasets are therefore a supplement to the real datasets.

All datasets are pre-processed and log-normalized. The procedure and whether this was performed by the authors or by author-guided tutorials are described in appendix A.

4.3 Pre-processing and Experiments

Pre-processing: We perform highly variable gene selection to reduce computational complexity as well as to align with standard analysis pipelines. Several methods exist to choose these genes. We use the *mean.var.plot* method from Seurat, which is the default in Scanpy. This method assumes log-normalized data. To investigate the impact of this step, we run experiments using both the default number of highly variable genes in Seurat (v5), namely 2000 and a substantially higher number of genes, namely 5000.

There does not appear to be a consensus in the field on whether genes (features) should be standardized to have zero mean and unit variance prior to PCA. Some argue

that the genes should be standardized (see for example the Seurat clustering tutorial) while others only center the genes prior to PCA (see for example the e-book single-cell best practices tutorial on dimensionality reduction and the scanpy clustering tutorial). The argument for standardizing genes is that it gives equal weight to all genes, preventing highly expressed genes from dominating the analysis. Contrary, differences in expression levels could contain biological signal, thus removing relevant information in the data by standardization. For comparison, we run both non-standardized and standardized versions. Note that some data-driven methods to determine the size of the PCA embedding assume that data is standardized.

Experiments: The purpose of the experiments is to compare different sizes (number of components) of the PCA embedding (low-dimensional representation of data). Directly assessing the quality of a PCA embedding is difficult, which makes such comparisons nontrivial. We measure the quality of a PCA embedding indirectly through the quality of the downstream clustering. We use the Leiden algorithm, as it is the default choice in both the Seurat and Scanpy pipelines. We choose this because clustering is one of the most common downstream tasks of scRNA-seq data. Additionally, for simulated data where ground truth is available, we assess embedding quality by measuring how well local neighborhood structure is preserved, using the trustworthiness metric.

We conduct two groups of similar experiments. The first group of experiments evaluate the quality of the PCA embeddings over a range of sizes. The steps are as follows:

1. Compute the PCA embedding of size k in the interval 1 to 1500, spaced such that

$$k = [2, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 200, 300, 400, 500, 750, 1000, 1500].$$

This gives rise to 18 different embeddings.

2. Compute trustworthiness if the ground truth expression data is available for the simulated data. This is the case for Symsim, Sparsim, and Splatter.
3. Cluster the data using the PCA embedding with the Leiden algorithm. See section C.1 for more information. The choice of hyper-parameters of the Leiden algorithm is discussed in section 4.5.
4. Measure the quality of the clustering using the adjusted rand index (ARI), which is described in detail in section C.2.

The second group of experiments mimics the first group of experiments, but evaluate the clustering quality obtained from PCA embeddings which size is determined by the data-driven methods. Thus, step one is changed slightly such that the steps are:

1a. Estimate the optimal number of components K by the data-driven methods described in section 4.1.

1aa. Measure the computation time for each method. Timings are obtained using the function *time.time()* from the time library in Python. All computations were run on a XeonE5-2650 processor with one core.

1b. Compute the PCA embedding of size K for each method.

All experiments were repeated 10 times using different seeds for the Leiden algorithm.

4.4 Analysis of the results

This section describes the evaluation criteria used to analyze the results of the experiments.

4.4.1 Clustering accuracy and stability

We compare the clustering quality obtained from the different PCA embeddings based on ranking of the ARI scores. For each dataset, we categorize the ARI scores into three

groups (good, adequate, bad) by comparing the ARI scores to thresholds based on ARI scores from random permutations of 10% and 25% of the labels. In the following, we describe this ranking in details.

Let $a_{r,m,\mathcal{D}}$ be the ARI score for data-driven method m on data set $\mathcal{D}$ and repetition r. The median of the ARI scores across all repetitions is denoted by $\bar{a}_{m,\mathcal{D}}$. These medians are compared to ARI scores from random permutations of the labels. Let $b_{p,s,\mathcal{D}}$ be the ARI scores obtained by the s'th permutation of p percent of the labels for dataset $\mathcal{D}$. Let $\bar{b}_{p,\mathcal{D}}$ be the median across s simulations. We use $s = 500$. The ranking of the ARI scores are then found by

$$\begin{aligned} \bar{a}_{m,\mathcal{D}} \geq \bar{b}_{10,\mathcal{D}} : &\qquad \text{good} \\ \bar{b}_{10,\mathcal{D}} > \bar{a}_{m,\mathcal{D}} \geq \bar{b}_{25,\mathcal{D}} : &\qquad \text{adequate} \\ \bar{b}_{25,\mathcal{D}} > \bar{a}_{m,\mathcal{D}} : &\qquad \text{bad} \end{aligned}$$

To summarize this into a measure per dataset we define three classes of clustering accuracy

$$\begin{aligned} \geq 60\% \text{ of all datasets rank as good} : &\qquad \text{Tier 1,} \\ \geq 40\% \text{ of all datasets rank as good} : &\qquad \text{Tier 2,} \\ < 40\% \text{ of all datasets rank as good} : &\qquad \text{Tier 3,} \end{aligned}$$

and three classes of clustering stability

$$\begin{aligned} < 15\% \text{ of all datasets rank as bad} : &\qquad \text{Tier 1,} \\ > 15\% \text{ of all datasets rank as bad} : &\qquad \text{Tier 2,} \\ > 33\% \text{ of all datasets rank as bad} : &\qquad \text{Tier 3.} \end{aligned}$$

4.4.2 Number of components

We categorize methods based on the estimated number of components such that

$$
\begin{aligned}
1 - 30 \text{ median number of components :} &\quad \text{Tier 1,} \\
31 - 150 \text{ median number of components :} &\quad \text{Tier 2,} \\
> 150 \text{ median number of components :} &\quad \text{Tier 3.}
\end{aligned}
$$

All things being equal, we prefer fewer components as this reduces complexity and downstream computational costs.

4.4.3 Neighborhood preservation

Neighborhood preservation was measured in terms of trustworthiness. We compute the median trustworthiness for each method and split these in three groups such that

$$
\begin{aligned}
\geq 0.7 \text{ median trustworthiness :} &\quad \text{Tier 1,} \\
\geq 0.4 \text{ median trustworthiness :} &\quad \text{Tier 2,} \\
< 0.4 \text{ median trustworthiness :} &\quad \text{Tier 3.}
\end{aligned}
$$

4.4.4 Convergence success rate

Methods that converged (i.e., estimated the number of components) for all datasets fall into Tier 1. If a method failed to converge for one or more datasets, it falls in Tier 3.

4.4.5 Scalability and speed

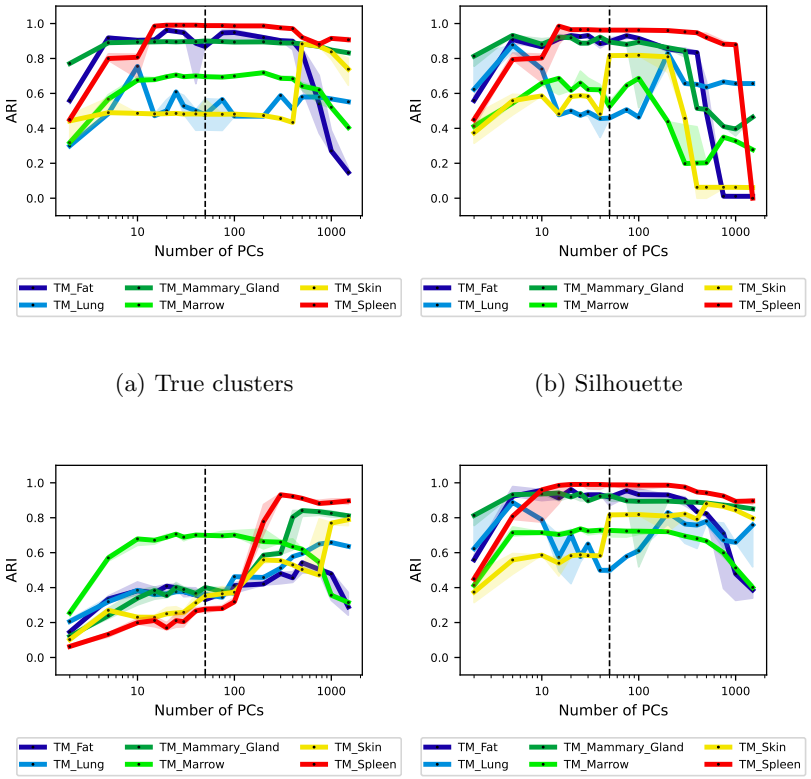
The scalability (slope) of all methods are similar, thus all methods are put in Tier 1.

The methods clearly split into two groups in terms of speed. KN solves a non-linear system of equations which may take several iterations to solve, and thus can "explode" in computation time. KN is therefore put in Tier 2 instead of Tier 1.

4.5 Methods to determine the hyper-parameters of the Leiden algorithm

The Leiden algorithm has two hyper-parameters, namely the number of neighbors which is used to construct the k-nearest-neighbor (KNN) graph and the resolution, which is the primary factor that determines the number of clusters [19]. We fix the number of neighbors to 15 which is the default value in Scanpy. The clustering quality and subsequent conclusions are highly sensitive to the number of Leiden clusters and thereby the resolution parameter as shown on the Tabula Muris datasets in fig. 9. The figure shows the ARI scores from four different methods to choose the resolution parameter. The methods denoted optimal and true clusters rely on extrinsic information to determine the resolution and are therefore generally not feasible in practice. However, they are useful when evaluating the quality of the embedding indirectly by clustering. The silhouette method uses only intrinsic information to determine the resolution and is thus, together with the default parameters, applicable in practice. The methods are described in detail below.

The ARI scores based on the true number of clusters and the silhouette scores resemble the largest ARI scores (optimal) with some fluctuations, while the ARI scores based on the default resolution are significantly lower for most datasets. Most important, notice that the peaks in cluster performance can be different depending on the choice of resolution parameter. Thus, conclusions on the quality of the PCA embedding may change depending on the choice of resolution.



(a) True clusters

(b) Silhouette

(c) Default

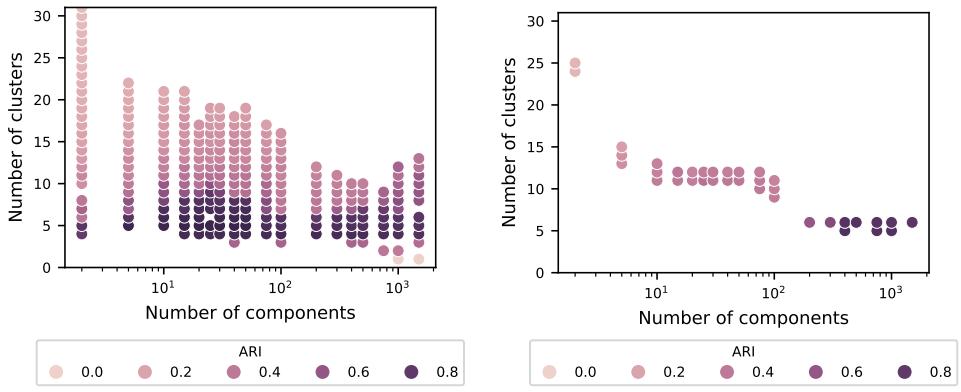
(d) Optimal

Fig. 9: Median ARI score for the Tabula Muris datasets using different settings of the resolution parameter (data is standardized but the pattern is similar for non-standardized). The shaded areas are the 80th percentile of the 10 repetitions. (a) controls the resolution such that the number of clusters is equal to the number of clusters in the true labels. (b) uses the maximal silhouette score to determine the resolution. (c) uses the default resolution in Scanpy of 0.8. (d) displays the resolution giving rise to the largest median ARI score across the 10 repetitions.

The most pronounced difference is seen for the default setting (fig. 9c), where we in contrast to the remaining settings would conclude that one should choose a large number of components for the datasets Spleen and Mammary Gland to gain optimal cluster performance. We find that this difference is caused by a dependence between

the number of clusters and the size of the PCA embedding keeping the resolution parameter fixed (see fig. 10). Concluding, that it is important to choose the resolution parameter in combination with the size of the PCA embedding in order to avoid bias in the results. This is in line with the findings in [19].

We will present results, where the resolution is determined by the silhouette scores (practically possible) and are aligned with the true clusters (not practically possible).



(a) Range of resolutions [0.05,1.5]

(b) Resolution 0.8

Fig. 10: Number of clusters obtained from different sizes of the PCA embedding (a) over the same range of resolutions between 0.05 and 1.5 and (b) resolution equal to 0.8 on the Tabula Muris Mammary Gland dataset (data is standardized). The colors indicate the ARI scores. If there are several observations we show the maximal ARI score.

4.5.1 Optimal

This method determines the resolution as the resolution which gives the largest median ARI score based on the 10 repetitions. This method reflects the maximal obtainable ARI scores for a particular embedding and is therefore relevant in this study.

4.5.2 True Clusters

This method determines the resolution such that the number of clusters is equal to the true number of clusters (in the labels). This means that it is possible that the resolution is slightly different over the 10 repetitions.

4.5.3 Silhouette

This method determines the resolution as the resolution which gives the largest median silhouette score [48] based on the 10 repetitions. The silhouette score measures the similarity of an observation to its own cluster compared to the neighboring cluster. The silhouette score ranges between -1 and 1 with a high score indicating that an observation is very similar to its own cluster but very dissimilar to the neighboring cluster. Let i be an observation assigned to cluster A, and let

$$a(i) = \frac{1}{|A| - 1} \sum_{j \in A, i \neq j} d(i, j)$$

be the average dissimilarity of observation i to all other observations in A, where $d(i, j)$ is the dissimilarity between observation i and j. Similarly, let $d(i, C)$ be the dissimilarity of observation i and cluster C and

$$b(i) = \min_{C \neq A} d(i, C)$$

be the average dissimilarity of observation i and all observations in the neighboring cluster B. The neighboring cluster B is thus the cluster which has the smallest average dissimilarity between observation i and all observations in that cluster. The silhouette score is then computed as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

if $|A| > 1$ and $s(i) = 0$ otherwise.

Note that the silhouette score has limitations (see eg [49] which describes limitation for the use in scRNA-seq data). Thus, in some scenarios the silhouette score may not reflect the optimal clustering.

4.5.4 Default

This method sets the resolution to be the default value in Scanpy v1.11, namely 0.8 (independent of data).

Supplementary information. This paper includes one additional file with the following specifications:

File name: Additional_file_1.pdf

Title: Appendix

Description: This file includes supplementary information to the manuscript "The necessity of data-driven optimization of the number of principal components in scRNA-seq analysis - examples on real data" in the form of text, figures, and tables.

Acknowledgements. This study was supported by DigitSTEM, an initiative funded by Bioneer A/S, a non-for-profit research organization.

Declarations

Ethics Approval

Not Applicable

Consent for Publication

Not Applicable

Availability of data and Materials

Links to the publicly available datasets supporting the conclusions of this article are included within the appendix section A.

All code is found on GitLab https://gitlab.com/digitstem/pca_components.git. For review, the code is uploaded to the journal.

Competing interests

The authors declare that they have no competing interests

Funding

This study was supported by DigitSTEM, an initiative funded by Bioneer A/S, a non-for-profit research organization.

Authors' contributions

MG conduction of the experiments, conceptualization of the study, implementation, analysis of the experiments and writing of the manuscript, AP supervised the study and revised the manuscript, LC supervised the study and revised the manuscript. All authors read and approved the final manuscript.

Appendix A Data

This section first describes the real scRNA-seq data from Tabula Muris (TM) [39], Tabula Sapiens (TS) [40], and the evaluation papers [3, 41] (EVAL), and then describes the simulated datasets from *Sparsim* (SPAR) [44] , *ScDesign2* (SCD) [45], *Splatter* (SPLAT) [46], and *Symsim* (SYMSIM) [47].

The tables in the next sections present characteristics of the datasets such as label (Label), number of classes (Classes), number of cells (Cells), number of genes (Genes), and sparsity (%) of the dataset when reduced to 2000 and 5000 highly variable genes

(Sparsity). If available, the datasets are downloaded including pre-processing (filtering, quality control) and log-normalization. If not available, we use provided tutorials to generate pre-processed and log-normalized data. The approach for each dataset is described below.

A.1 Tabula Muris (TM)

The Tabula Muris project [39] provides an atlas of scRNA-seq datasets from 20 organs and tissues from Mus Musculus. We use a subset of all available datasets and present the characteristics of these in table A1. The Tabula Muris (TM) data are downloaded from Figshare[2]. We downloaded version two of the scRNA-seq data from Smart-seq2 sequencing of FACS sorted cells. The downloaded data is the raw counts. We used the tutorial provided by the authors to pre-process the data and used the cell ontology class as labels. The tutorial can be found in this Github repo. The number of cells and genes reported in table A1 reflects values after pre-processing. Some classes have a very small number of observations, compared to all other classes. We remove all classes where the number of observations in the class is less than 1% of data and indicate this by a star (∗) in the column "notes" in table A1.

Table A1: Summary of the Tabula Muris datasets. Parenthesis indicate the actual number after applying the action(s) indicated in the column "notes".

Organ	Label	Classes	Cells	Genes	Sparsity	Notes
Fat	cell type	7	4709	18297	89/87	
Lung	cell type	12	1620	16496	89/89	
Mammary_Gland	cell type	4	2304	16618	85/84	
Marrow	cell type	22 (16)	4897 (4710)	17479	84/83	∗
Skin	cell type	5 (4)	2263 (2248)	16180	82/82	∗
Spleen	cell type	3	1689	14600	90/90	

[2] https://figshare.com/projects/Tabula_Muris_Transcriptomic_characterization_of_20_organs_and_tissues_from_Mus_musculus_at_single_cell_resolution/27733

A.2 Tabula Sapiens (TS)

The Tabula Sapiens project [40] is a human cell atlas consisting of scRNA-seq datasets from several organs and tissues from Homo Sapiens. We use a subset of all available data and present the characteristics of these in table A2. The Tabula Sapiens (TS) data are downloaded from Figshare[3]. We use the log-normalized counts and the cell ontology class as labels. Some classes have a very small number of observations, compared to all other classes. We remove all classes where the number of observations in the class is less than 1% of data and indicate this by a star (∗) in the column "notes" in table A2.

Table A2: Summary of the Tabula Sapiens datasets. Parenthesis indicate the actual number after applying the action(s) indicated in the column "notes".

Organ	Label	Classes	Cells	Genes	Sparsity	Notes
Ear	cell type	16 (9)	3055 (3029)	61806	90/92	∗
Kidney	cell type	13 (6)	11376 (11148)	61806	89/92	∗
Large Intestine	cell type	29 (16)	30084 (28882)	61,806	94/94	∗
Liver	cell type	24 (15)	22214 (21531)	61806	95/95	∗
Pancreas	cell type	23 (8)	14140 (13758)	61806	91/91	∗
Prostate	cell type	14 (9)	21030 (20634)	61806	91/92	∗
Testis	cell type	15 (3)	7513 (7195)	61806	89/91	∗
Trachea	cell type	28 (13)	22671 (21969)	61806	92/92	∗

A.3 Evaluation Papers (EVAL)

The datasets from the evaluation papers [3, 41] are collected from different sources.

The Chubatch datasets [50] consists of snapshots of progenitor cells collected from human pluripotent stem cells in two different batches (batch 1 and batch 2). The chubatch datasets are also downloaded from the NCBI gene expression omnibus[4]

[3] https://figshare.com/articles/dataset/Tabula_Sapiens_v2/27921984
[4] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE75748

[Accession number: GSE75748]. This data is pre-processed and normalized but not logarithmized. We therefore logarithmize these two datasets.

The Koh paper [51] studies human mesoderm development and collects single cell data (using Fluidigm C1) across several cell types. The Kumar paper [52] studies pluripotent stem cells (PSC) in mice under different growth conditions while the zheng datasets (Zheng4eq, Zheng8eq, Zheng4uneq) were assembled by [41] from subsets of the data collected by [53]. The authors of [53] studies a droplet based system to count 3'mRNA in celll samples and sequence several thousands of cells by the 10X genomics platform. The Koh, Kumar, Zheng4eq, Zheng8eq, Zheng4uneq datasets are retrieved from the R package *DuoClustering2018* under the bioconductor initiative[5]. For these datasets we save the full datasets of log-normalised counts found in the assays of the SingleCellExperiment object under the name *logcounts()*.

The mixology datasets [54] consists of 7 single-cell datasets downloaded from Github[6]. These are scRNA-seq benchmarking datasets according to the authors and consist of human lung adenocarcinoma cell lines. Four of the datasets contain the pre-processed and log-normalized counts in the layer *logcounts()* (see the tutorial[7] for normalization of data). In three of the datasets (sc_Celseq2_5cl_p1, sc_Celseq2_5cl_p2, and sc_Celseq2_5cl_p3) the log-normalized counts are missing and we have not been able to determine the normalization method. We will therefore not consider these datasets.

A summary of the datasets are found in table A3. Note that the mixology datasets are found in [3] under the name FreytagGold.

A.4 Sparsim (SPAR)

Sparsim simulates scRNA-seq data based on a Gamma-Multivariate Hypergeometric distribution. This simulation method allows for estimation of simulation parameters

[5] https://bioconductor.org/packages/release/data/experiment/html/DuoClustering2018.html
[6] https://github.com/LuyiTian/sc_mixology?tab=readme-ov-file
[7] https://github.com/LuyiTian/sc_mixology/blob/master/script/normalization_imputation

Table A3: Summary of the datasets from evaluation papers (EVAL).

Name	Label	Classes	Cells	Genes	Sparsity	Reference
chubatch1	cell type	5	350	19099	63/62	[50]
chubatch2	cell type	6	425	19099	65/62	[50]
koh	cell type	9	531	48981	69/77	[51]
kumar	growth condition	3	246	45159	40/41	[52]
zheng4eq	cell type	4	3994	15568	97/97	[53]
zheng8eq	cell type	8	3994	15716	97/97	[53]
zheng4uneq	cell type	4	6,498	16443	96/97	[53]
10x_qc	cell line	3	902	16468	52/49	[54]
10x_5cl_qc	cell line	5	3918	11786	69/67	[54]
CELseq2_qc	cell line	3	274	28204	78/78	[54]
Dropseq_qc	cell line	3	225	15127	71/67	[54]

from real scRNA-seq datasets. We use the datasets listed in table A4 to generate the simulated data. All datasets have been reduced to only include the 5000 highest variable genes (using the *mean.var.plot* method in Seurat which corresponds to the default method in Scanpy) before given as input to the simulator. We simulate the same number of cells and the same group distribution (imbalanced) as in the real scRNA-seq datasets. In order to investigate the impact of imbalance in the cell groups we additionally simulate balanced data. The simulation code is located in the folder *simulate_data* at the GitLab repository (see link in Availability of data and Materials).

A.5 ScDesign2 (SCD)

ScDesign2 chooses a probabilistic model for each gene (either Poisson, Negative binomial or their zero-inflated counter-parts) in a data-driven manner and then models the correlation between genes by a Gaussian copula model to simulate scRNA-seq data. Thus, ScDesign2 uses real scRNA-seq data to build the model from which it simulates scRNA-seq datasets. We use the datasets listed in table A4 to generate the simulated data. All datasets have been reduced to only include the 5000 highest variable genes

(using the *mean.var.plot* method in Seurat which corresponds to the default method in Scanpy) before given as input to the simulator. We simulate the same number of cells and the same group distribution (imbalanced) as in the real scRNA-seq datasets. In order to investigate the impact of imbalance in the cell groups we additionally simulate balanced data. The simulation code is located in the folder *simulate_data* at the GitLab repository (see link in Availability of data and Materials).

A.6 Splatter (SPLAT)

Simulations from Splatter are based on a gamma-Poisson distribution. Splatter does not allow for parameter estimation for multiple cell groups. Instead, we estimate the parameters for the gamma distribution and expression outlier using one of the cell types in the real scRNA-seq datasets. The remaining parameters are kept as default. All datasets have been reduced to only include the 5000 highest variable genes (using the *mean.var.plot* method in Seurat which corresponds to the default method in Scanpy) before given as input to the simulator. We simulate the same number of cells and the same group distribution (imbalanced) as in the real scRNA-seq datasets. In order to investigate the impact of imbalance in the cell groups we additionally simulate balanced data. We use the datasets listed in table A4 to generate the simulated data. The simulation code is located in the folder *simulate_data* at the GitLab repository (see link in Availability of data and Materials).

A.7 Symsim (SYMSIM)

Symsim is an R-package which simulates scRNA-seq data in a two-step process. First, it simulates the biological variance (true counts) followed by a simulation of the technical variance which is added hereto to obtain the observed counts. Symsim does not allow for estimation of simulation parameters from real datasets, but has the advantage that it can simulate a hierarchical structure of the cell types.

Table A4: Summary of the real scRNA-seq datasets used to estimate parameters for the simulation of data from Sparsim, ScDesign2, and Splatter.

Name	**Classes**	**Cells**	**Genes**	**Sparsity** (SPAR)	**Sparsity** (SCD)	**Sparsity** (SPLAT)	**Source**
Ear	9	3029	5000	95/93	93/92	36/36	TS
Kidney	6	11148	5000	92/88	93/87	19/19	TS
Pancreas	8	13758	5000	93/91	93/90	51/51	TS
Testis	3	7494	5000	92/89	91/88	38/38	TS
Lung	12	1620	5000	91/88	92/89	15/15	TM
Mammary_Gland	4	2304	5000	88/84	89/84	19/19	TM
Marrow	16	4710	5000	86/	85/81	12/12	TM
Spleen	3	1689	5000	92/89	92/90	13/13	TM
Dropseq_qc	3	225	5000	77/75	71/67	23/23	EVAL
koh	9	531	5000	86/82	81/75	51/51	EVAL
kumar	9	531	5000	71/74	52/44	34/34	EVAL
zheng4eq	4	3,994	5000	97/97	96/96	24/24	EVAL

We simulate data by Symsim with 8 cell groups (see fig. A1), 5000 cells and 5000 genes (see table A5). This simulated data is balanced, since the simulation engine does not allow for different group sizes except for one group. The parameter settings for the Symsim functions can be found in the code located in the folder *simulate_data* at the GitLab repository (see link in Availability of data and Materials).

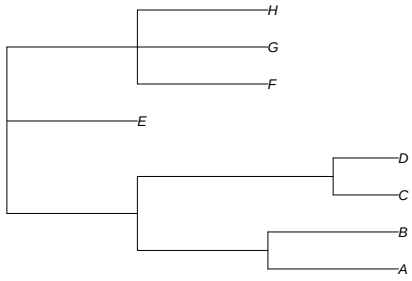


Fig. A1: Cell group (class) structure of the simulated data.

Table A5: Summary of the Symsim dataset.

Name	Classes	Cells	Genes	Sparsity
SYMSIM	8	5000	5000	

Appendix B Algorithm

Algorithm 1 Jackstraw

Set the parameters: $f = 0.01^*$, $R = 100^*$, $d = \min(k_{\max}, 500)$, $\tau_{JS} = 0.00001^*$, $\tau_{\text{JS_score}} = 0.05$, $k_{\max} = \text{round}(q - 0.1q)$. A star indicates that it is the same as the default in Seurat.
Compute loadings $\mathbf{V} \in \mathbb{R}^{q \times p}$ of the standardized data $\mathbf{X}_{\text{std}} \in \mathbb{R}^{n \times p}$

// Simulate loadings under H_0 (no association between the j'th gene and the i'th principal component //
for $r = 1, \ldots, R$ **do**
    Randomly select a subset of the genes, and denote this subset $\mathbf{g}_{\text{subset}} \in \mathbb{R}^s$ where $s = fp$
    Permute the columns of $\mathbf{X}_{\text{std}}$ corresponding to $\mathbf{g}_{\text{subset}}$ to create the adjusted dataset $\tilde{\mathbf{X}}_{\text{std}}$
    Compute loadings $\tilde{\mathbf{V}}$ of the adjusted dataset $\tilde{\mathbf{X}}_{\text{std}}$
    Let $\tilde{\mathbf{V}}_{\text{subset}} = \tilde{\mathbf{V}}[:, \mathbf{g}_{\text{subset}}] \in \mathbb{R}^{q \times s}$ to extract the null-loadings
end for
Let $\tilde{\mathbf{V}}_{\text{all subsets}} \in \mathbb{R}^{q \times s \times R}$ be the collection of null-loadings
// Obtain empirical p-values for the association between the j'th gene and the i'th principal component //
for $i = 1, \ldots, q$ **do**
    for $j = 1, \ldots, p$ **do**
        Set $V_{\text{ref}} = \mathbf{V}[i, j]$
        Compute the empirical p-values for the association between the j'th gene and the i'th principal component

$$\mathbf{p}[i, j] = \frac{1}{Rs} \sum_{r=1}^{R} \sum_{l=1}^{s} \mathbb{1}\left(V_{\text{ref}} \leq \tilde{\mathbf{V}}_{\text{all subsets}}[i, l, r]\right),$$

        where $\mathbb{1}()$ denotes the indicator function.
    end for
end for
// Obtain Jackstraw scores of the association between the i'th principal component and the genes //
for $i = 1, \ldots, d$ **do**
    Compute the actual number of empirical p-values below the threshold τ_{JS}, namely $n_{\text{below threshold}} = \sum_{j=1}^{p} \mathbb{1}(\mathbf{p}[i, j] \leq \tau_{\text{JS}})$
    if $n_{\text{below}} = 0$ **then**
        $\mathbf{J}[i] = 1$ by definition
    else
        Perform a binomial test with n_{below} successes, p trials and a probability of τ_{JS}
        The resulting p-value from the binomial test is the jackstraw score $\mathbf{J}[i]$
    end if
end for
$K = \arg\min\{\mathbf{J} > \tau_{\text{JS_score}}\}$

Appendix C Pre-processing and Experiments

C.1 The Leiden algorithm

Clustering using the Leiden algorithm is performed by the functions *sc.pp.neighbors* and *sc.tl.leiden* from the Python library Scanpy (v1.11) [6]. The former function builds the nearest-neighbor (NN) graph, while the preceding function applies the Leiden algorithm to obtain the clustering. The hyper-parameters are the number of neighbors $n_{\text{neighbors}}$ to build the NN-graph and the resolution r of the Leiden algorithm.

C.2 Clustering Metrics

We evaluate the cluster quality based on adjusted rand index (ARI). We compute adjusted rand index by the Python package scikit-learn (v 1.8) [55] using the function *adjusted_rand_score()*.

The rand index (RI) [56] measures the ratio between pairs of observations in agreement and the total number of pairs in data. A pair of observations is in agreement if they belong to the same class and end up in the same cluster or belong to different classes and end up in different clusters. That is,

$$\text{RI} = \frac{n_{\text{agreement}}}{n_{\text{pairs}}},$$

where $n_{\text{agreement}}$ is the number of pairs in agreement and n_{pairs} is the total number of pairs.

The adjusted rand index (ARI) is adjusted for chance and given by

$$\text{ARI} = \frac{\text{RI} - \text{Expected RI}}{\text{Maximum RI} - \text{Expected RI}}.$$

C.3 Neighborhood preservation

We measure the degree to which an embedding preserve neighborhoods using the trustworthiness measure. The trustworthiness is computed by the python package scikit-learn (v 1.8) [55] using the function *trustworthiness()*. For each observation i the trustworthiness compares the k nearest neighbors of the output space to the neighbors of the input space. It is computed as

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i^k} \max 0, r(i,j) - k$$

where $\mathcal{N}_i^k$ are the k nearest neighbors of observation i in output space and $r(i,j)$ is the rank in terms of nearest neighbor that observation j in input space has to observation i in output space (see function documentation in scikit-learn [55]).

Appendix D Results

D.1 Performance profiles

Real scRNA-seq data, standardized, and 2000 HVG

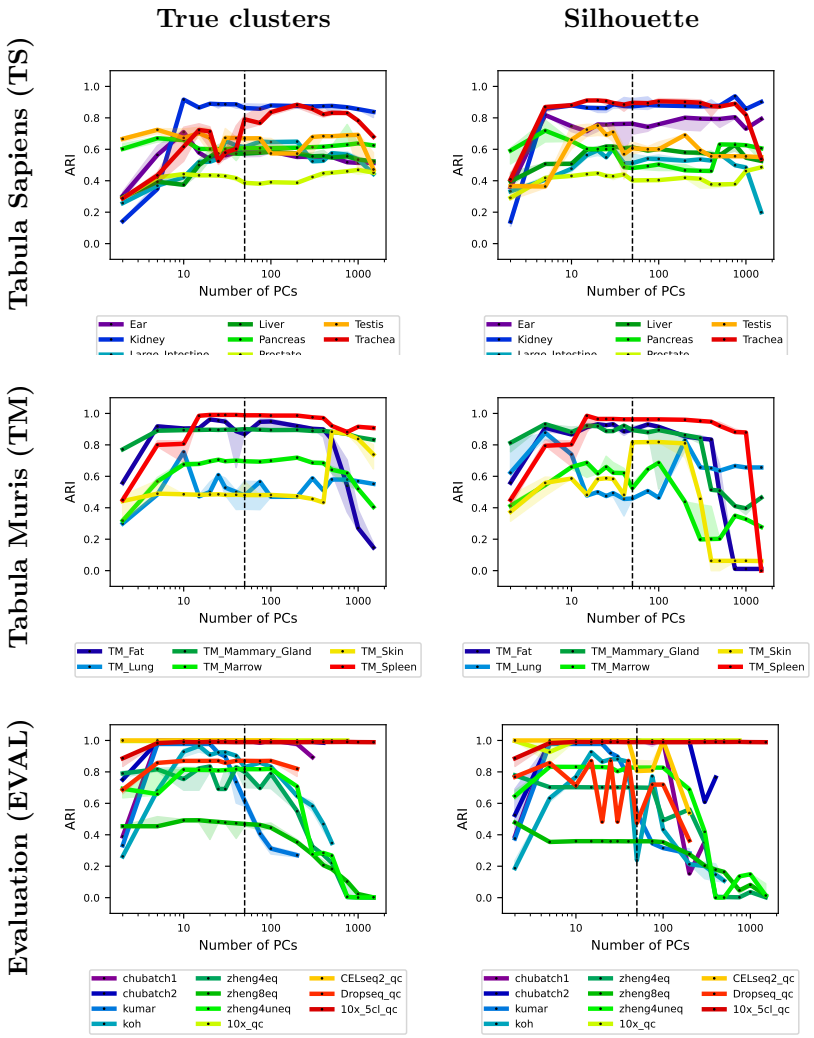


Fig. D2: Performance profiles (ARI scores) of the real scRNA-seq datasets over a range of sizes of the PCA embedding. Left column uses a resolution for the Leiden algorithm aligned with the true clusters while the right column choose the resolution based on the silhouette score. These figures use 2000 highly variable genes and data is standardized.

Real scRNA-seq data, standardized, and 5000 HVG

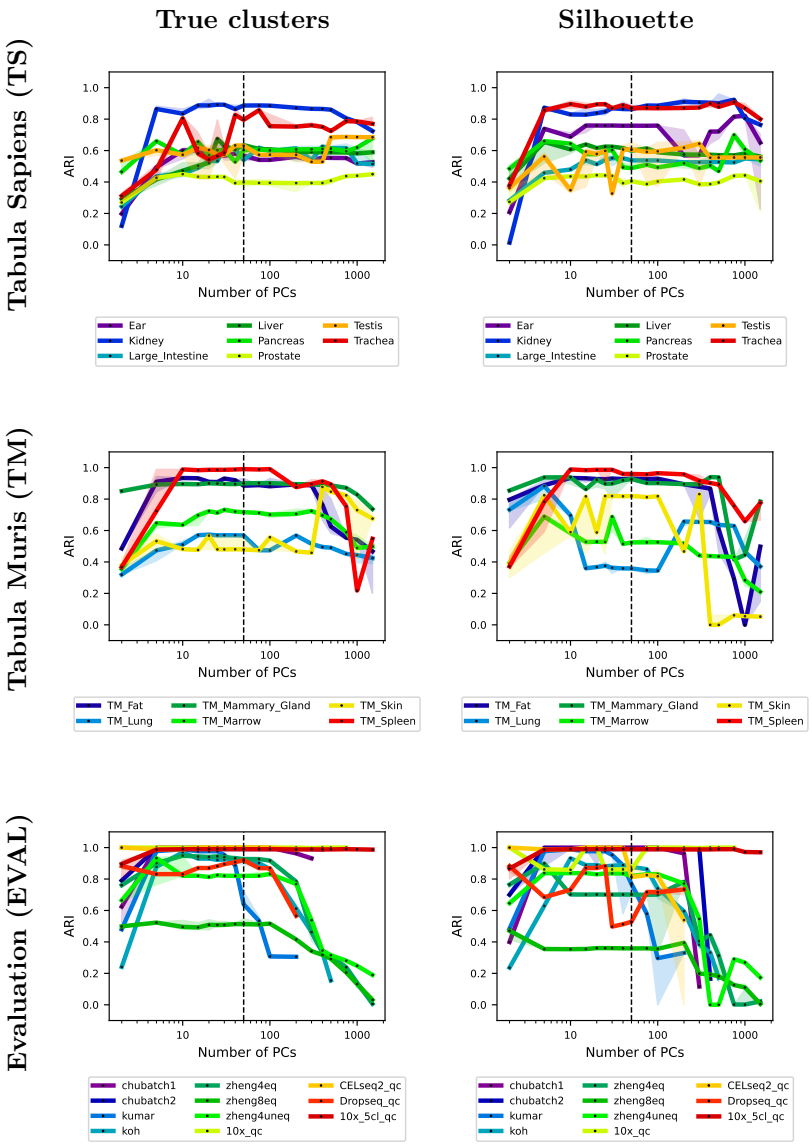


Fig. D3: Performance profiles (ARI scores) of the real scRNA-seq datasets over a range of sizes of the PCA embedding. Left column uses a resolution for the Leiden algorithm aligned with the true clusters while the right column choose the resolution based on the silhouette score. These figures use 5000 highly variable genes and data is standardized.

Real scRNA-seq data, non-standardized, and 2000 HVG

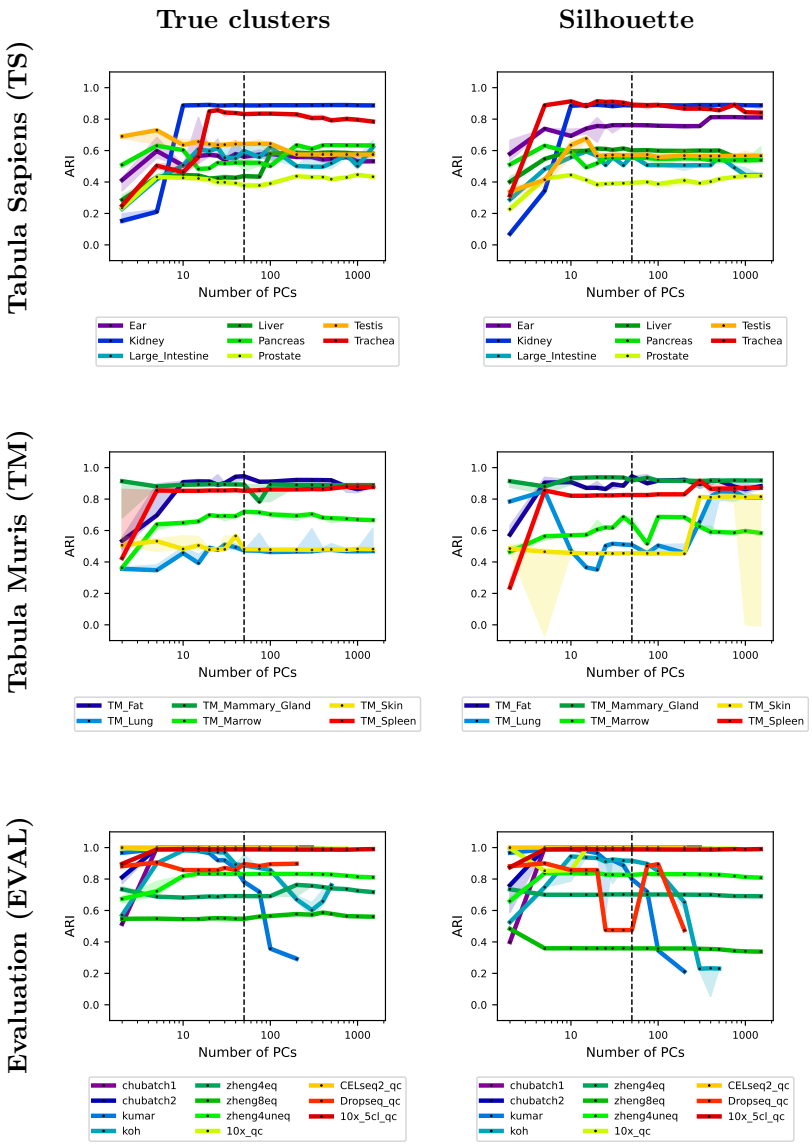


Fig. D4: Performance profiles (ARI scores) of the real scRNA-seq datasets over a range of sizes of the PCA embedding. Left column uses a resolution for the Leiden algorithm aligned with the true clusters while the right column choose the resolution based on the silhouette score. These figures use 2000 highly variable genes and data is non-standardized.

Simulated scRNA-seq data, standardized, and 2000 HVG

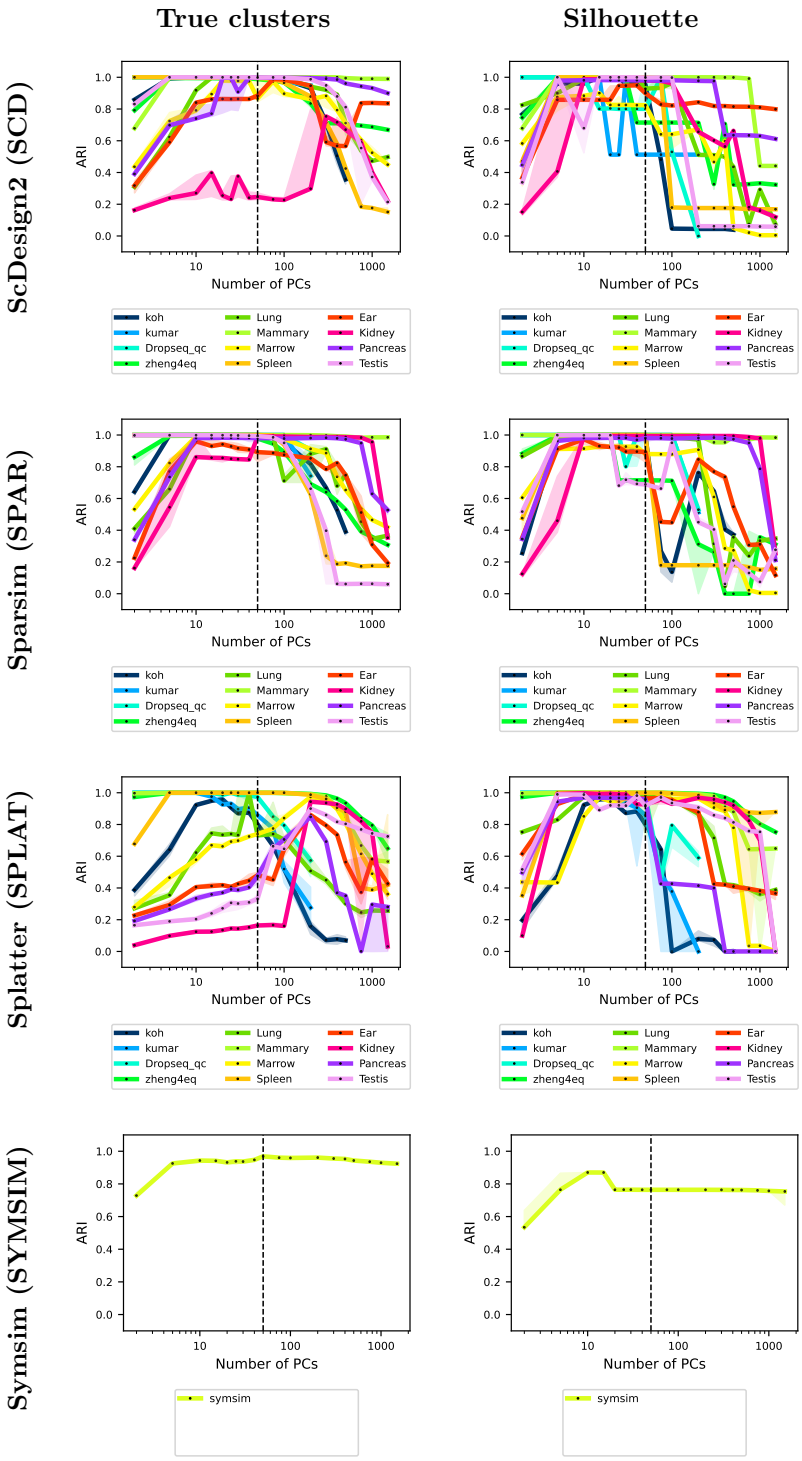


Fig. D5: Performance profiles (ARI scores) of the simulated scRNA-seq datasets over a range of sizes of the PCA embedding. Left column uses a resolution for the Leiden algorithm aligned with the true clusters while the right column choose the resolution based on the silhouette score. These figures use 2000 highly variable genes and data is standardized.

Simulated scRNA-seq data, standardized, and 5000 HVG

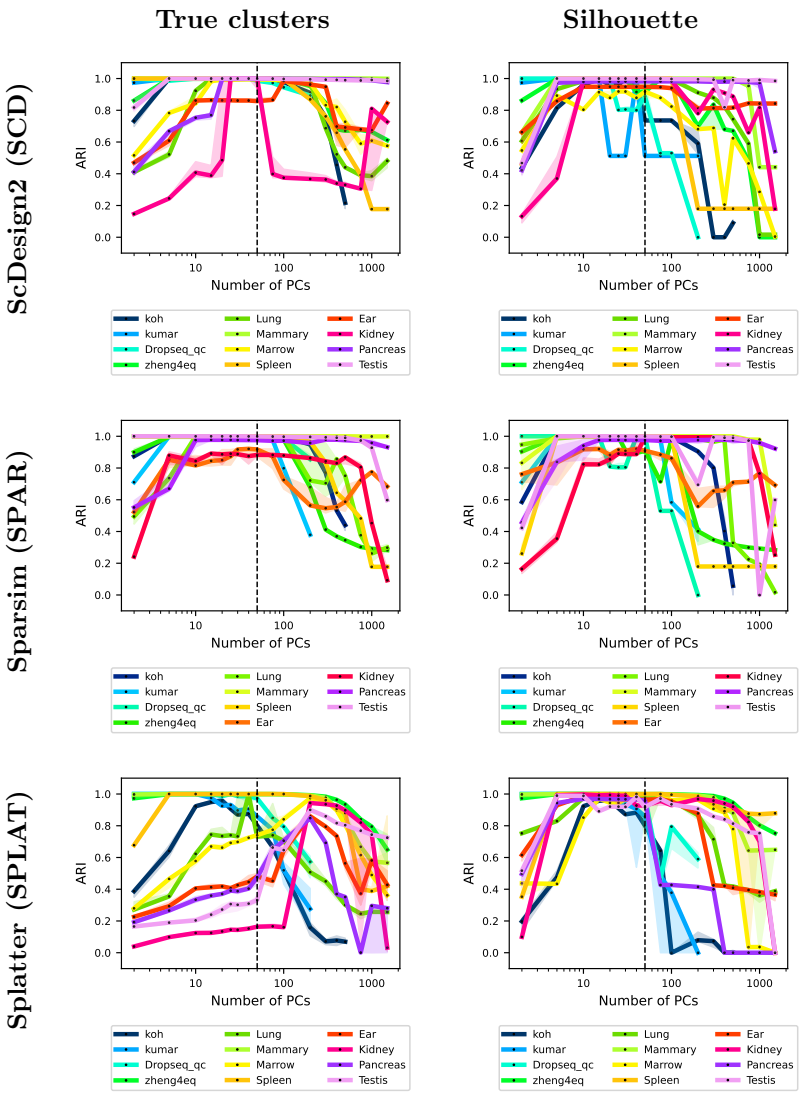


Fig. D6: Performance profiles (ARI scores) of the simulated scRNA-seq datasets over a range of sizes of the PCA embedding. Left column uses a resolution for the Leiden algorithm aligned with the true clusters while the right column choose the resolution based on the silhouette score. These figures use 5000 highly variable genes and data is standardized. Symsim and the Sparsim Marrow dataset did not finish within 72 hours using 5000 highly variable genes.

Simulated scRNA-seq data, non-standardized, and 2000 HVG

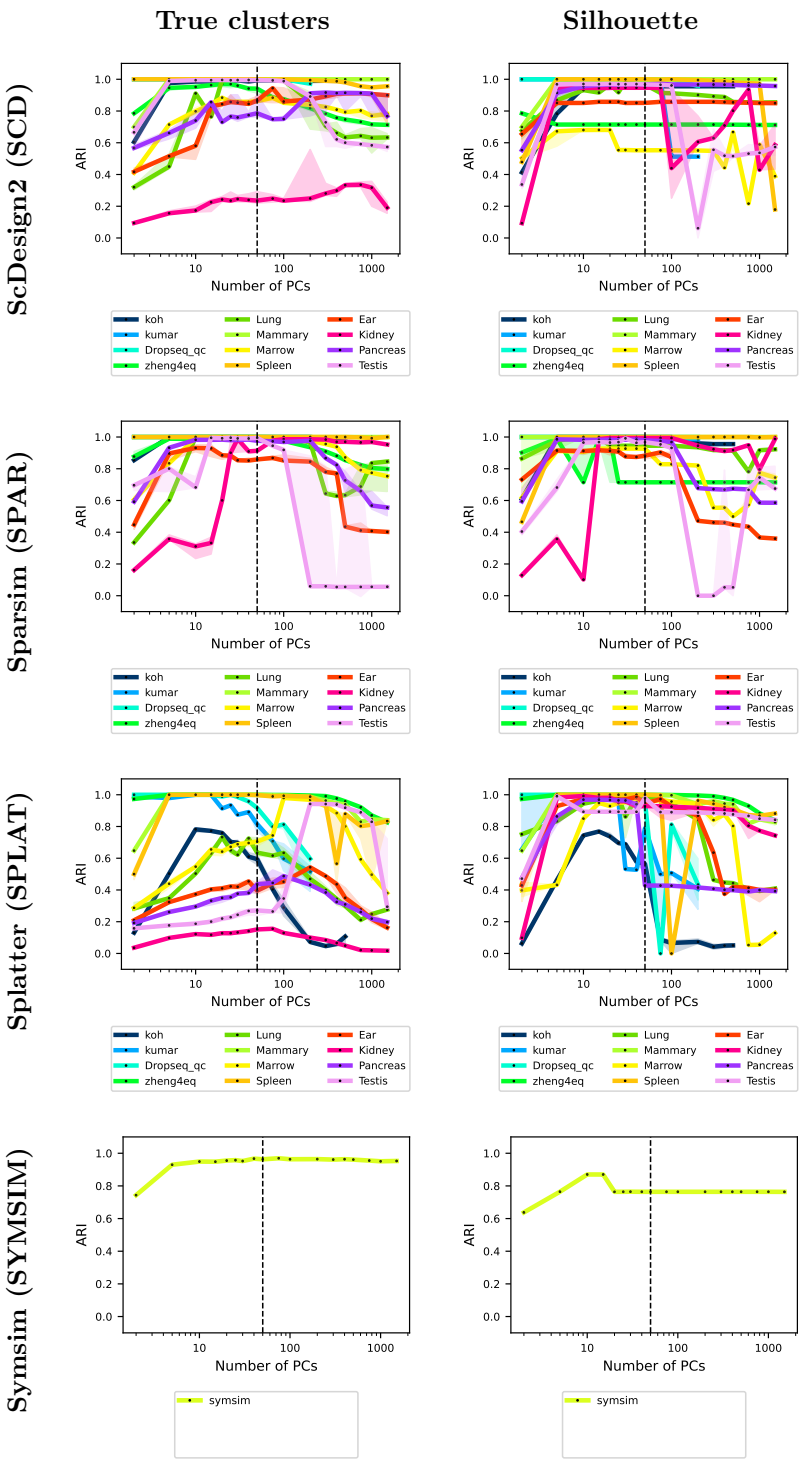


Fig. D7: Performance profiles (ARI scores) of the simulated scRNA-seq datasets over a range of sizes of the PCA embedding. Left column uses a resolution for the Leiden algorithm aligned with the true clusters while the right column choose the resolution based on the silhouette score. These figures use 2000 highly variable genes and data is non-standardized.

Simulated scRNA-seq data with balanced groups, standardized, and 2000 HVG

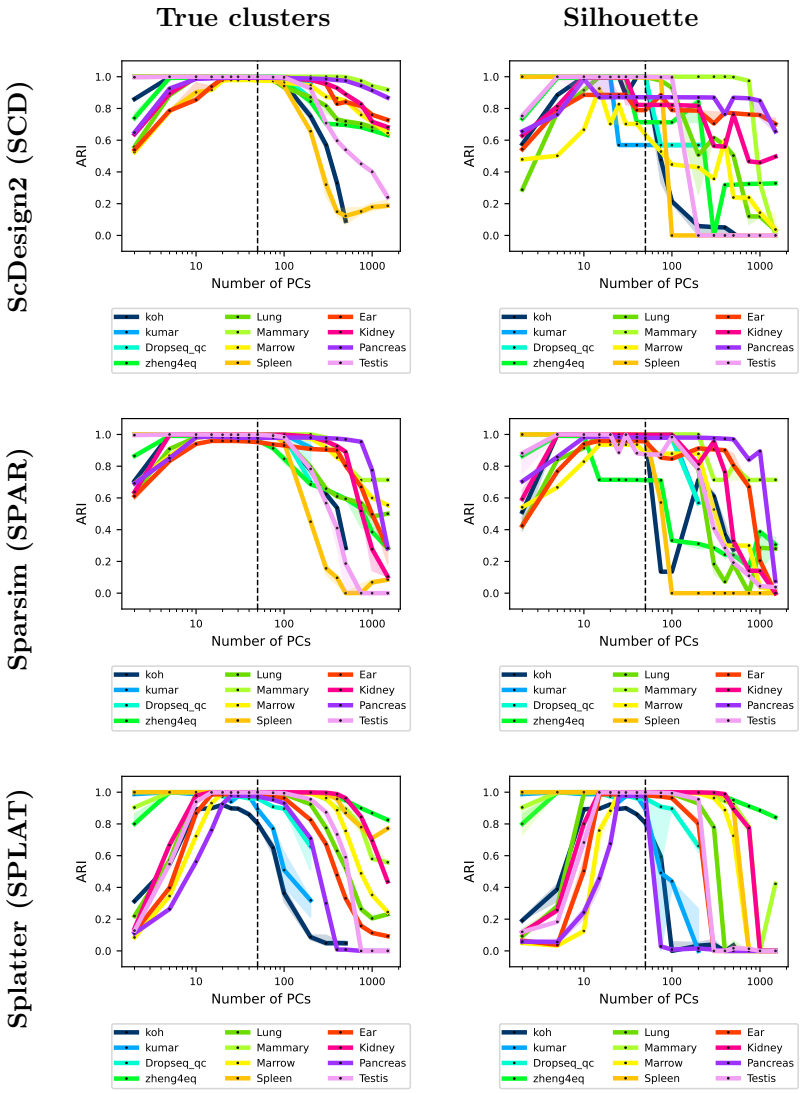
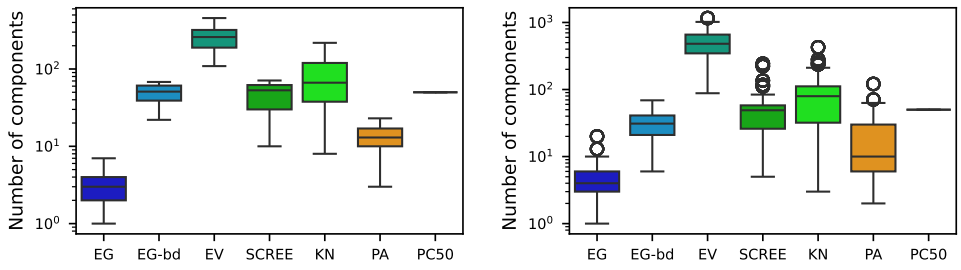


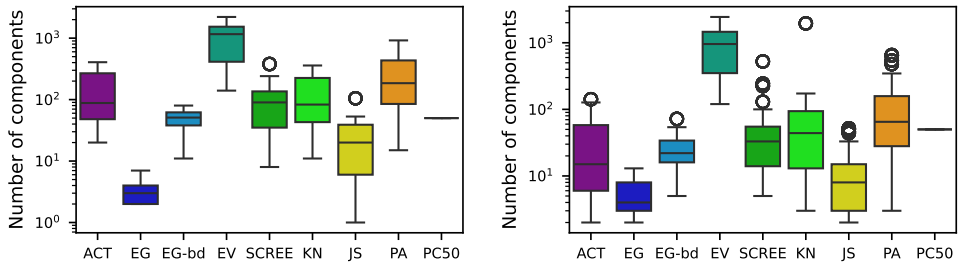
Fig. D8: Performance profiles (ARI scores) of the simulated scRNA-seq datasets over a range of sizes of the PCA embedding. Left column uses a resolution for the Leiden algorithm aligned with the true clusters while the right column choose the resolution based on the silhouette score. These figures use simulated data with balanced groups and use 2000 highly variable genes and data is standardized.

D.2 Number of components



(a) Real scRNA-seq data. (b) Simulated scRNA-seq data.

Fig. D9: Boxplots on a log-scale of the number of components estimated by each method. These figures do not standardize data before applying PCA and use 2000 highly variable genes.



(a) Real scRNA-seq data. (b) Simulated scRNA-seq data.

Fig. D10: Boxplots on a log-scale of the number of components estimated by each method. These figures standardize data before applying PCA and use 5000 highly variable genes.

D.3 Heatmaps of clustering performance

Resolution aligned with true number of clusters

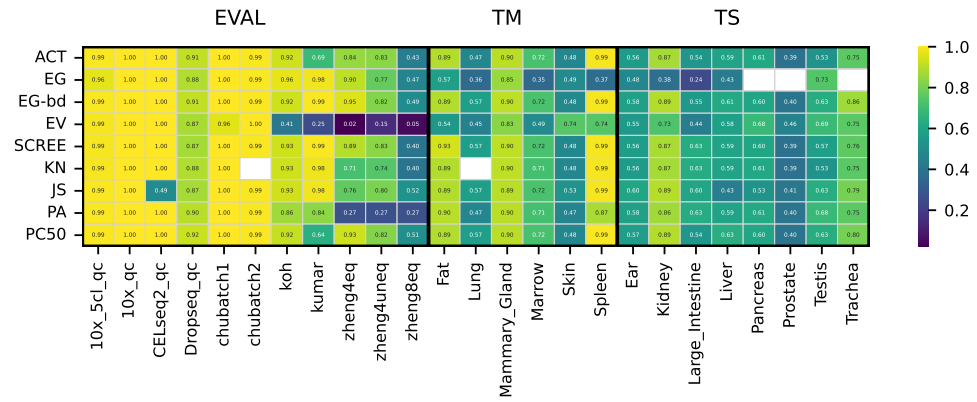


Fig. D11: Heatmap of the ARI scores across the data-driven methods for each real scRNA-seq dataset. The analysis standardizes data before applying PCA and use 5000 highly variable genes.

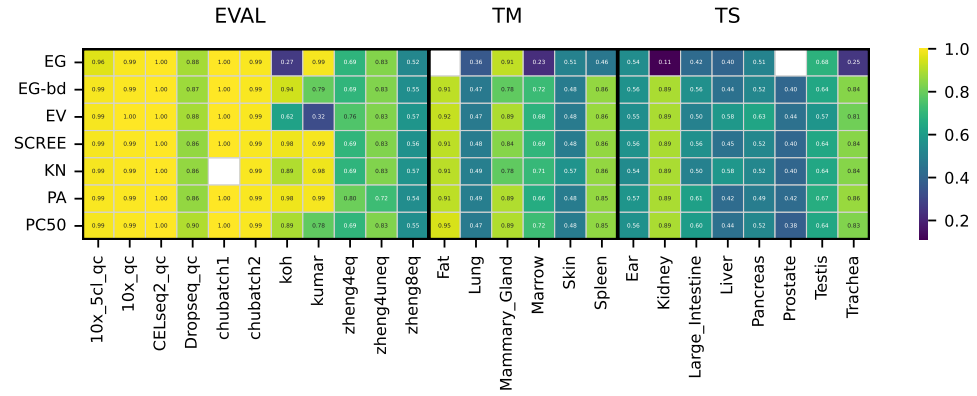


Fig. D12: Heatmap of the ARI scores across the data-driven methods for each real scRNA-seq dataset. The analysis does not standardize data before applying PCA and use 2000 highly variable genes.

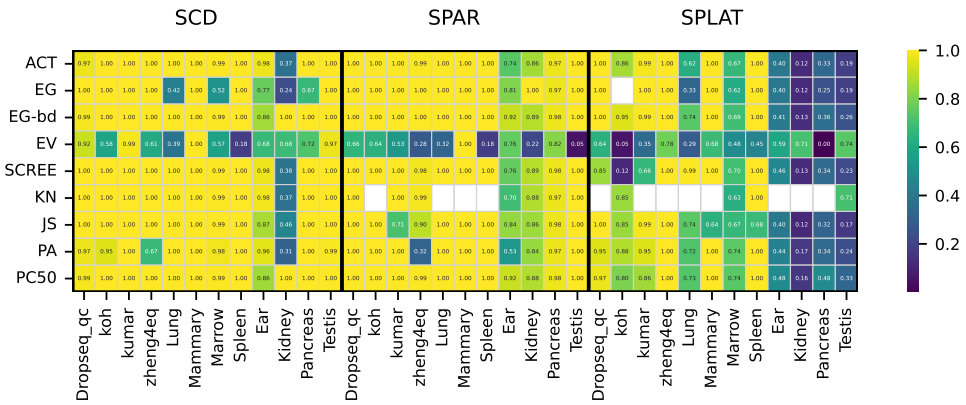


Fig. D13: Heatmap of the ARI scores across the data-driven methods for each simulated scRNA-seq dataset. The analysis standardizes data before applying PCA and use 5000 highly variable genes.

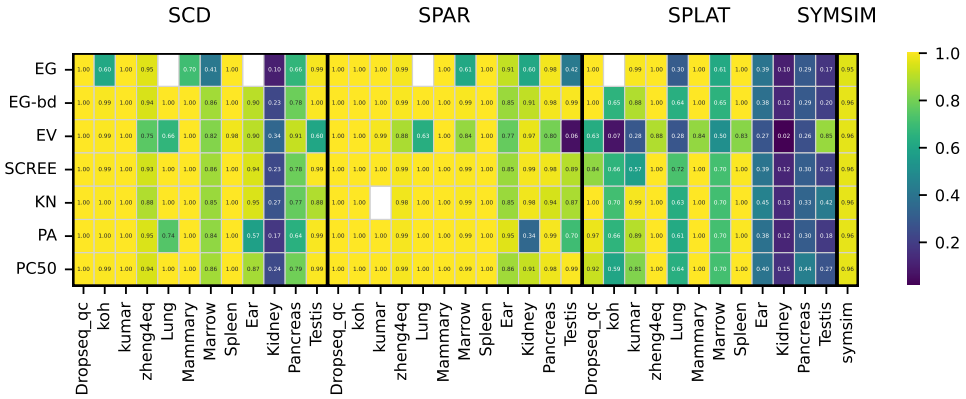


Fig. D14: Heatmap of the ARI scores across the data-driven methods for each simulated scRNA-seq dataset. The analysis does not standardize data before applying PCA and use 2000 highly variable genes.

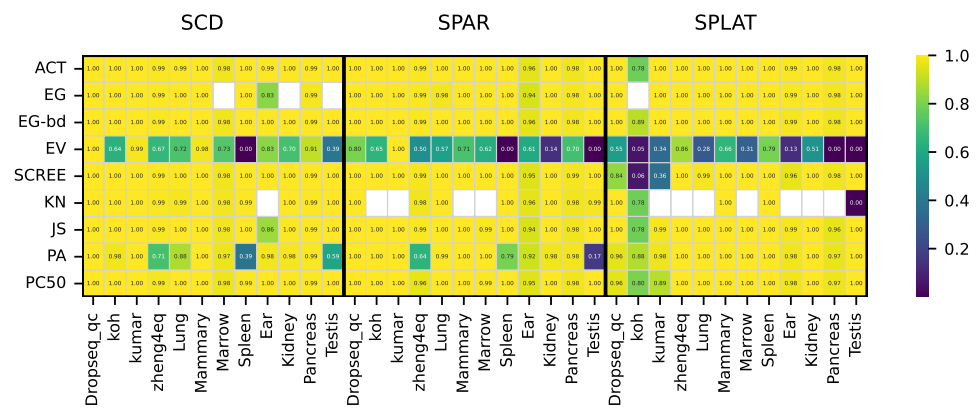


Fig. D15: Heatmap of the ARI scores across the data-driven methods for each simulated scRNA-seq dataset. The analysis standardize data before applying PCA and use 2000 highly variable genes. The simulated data is constructed to have balanced clusters.

Resolution based on silhouette score

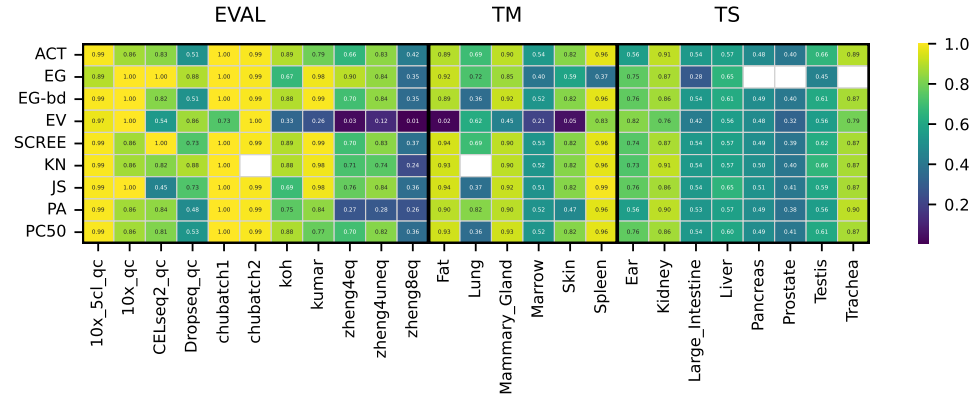


Fig. D16: Heatmap of the ARI scores across the data-driven methods for each real scRNA-seq dataset. The analysis standardizes data before applying PCA and use 5000 highly variable genes.

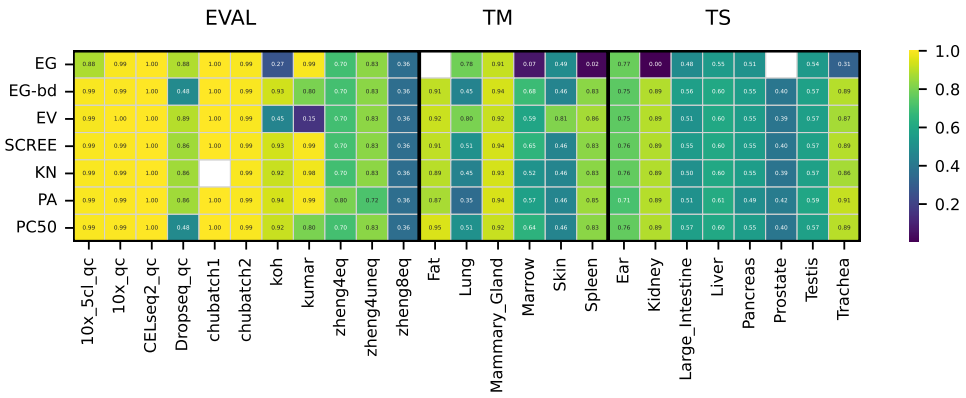


Fig. D17: Heatmap of the ARI scores across the data-driven methods for each real scRNA-seq dataset. The analysis does not standardize data before applying PCA and use 2000 highly variable genes.

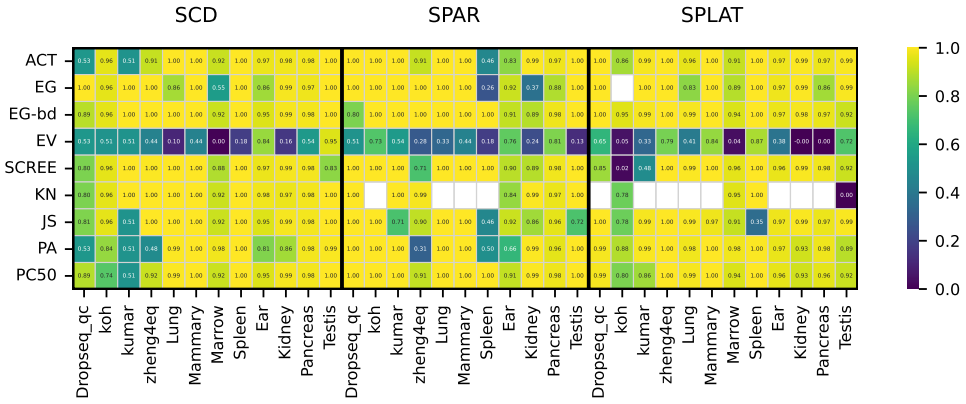


Fig. D18: Heatmap of the ARI scores across the data-driven methods for each simulated scRNA-seq dataset. The analysis standardizes data before applying PCA and use 5000 highly variable genes.

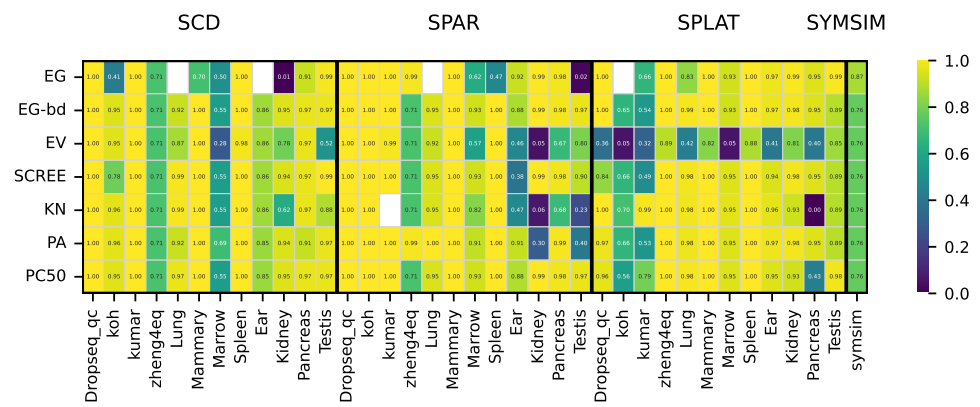


Fig. D19: Heatmap of the ARI scores across the data-driven methods for each simulated scRNA-seq dataset. The analysis does not standardize data before applying PCA and use 2000 highly variable genes.

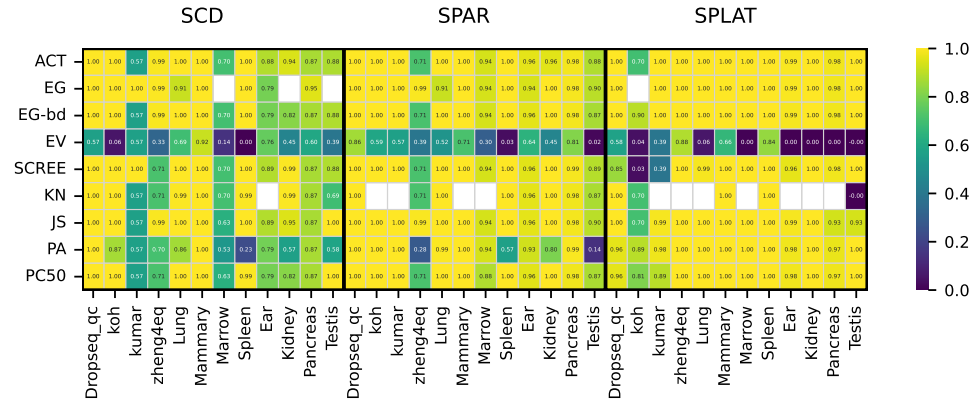


Fig. D20: Heatmap of the ARI scores across the data-driven methods for each simulated scRNA-seq dataset. The analysis standardize data before applying PCA and use 2000 highly variable genes. The simulated data is constructed to have balanced clusters.

D.4 Heatmaps of trustworthiness scores

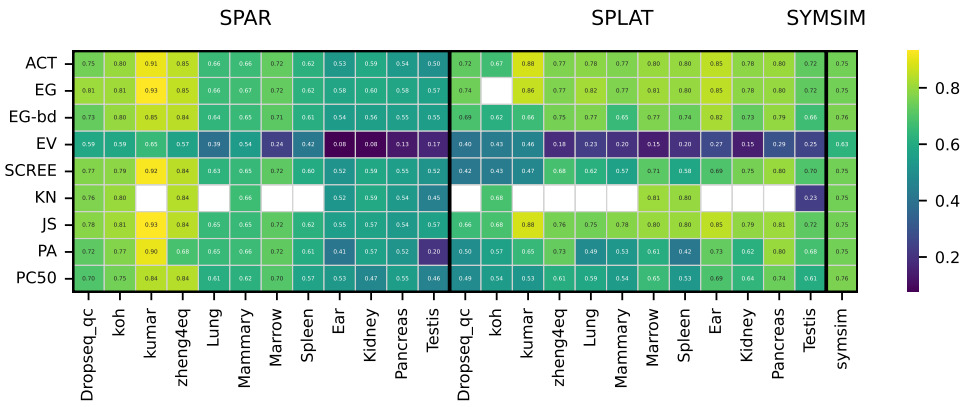


Fig. D21: Heatmap of the trustworthiness using 50 neighbors on each simulated scRNA-seq dataset where there exist a ground truth for each method.

Standardized data, and 5000 HVG

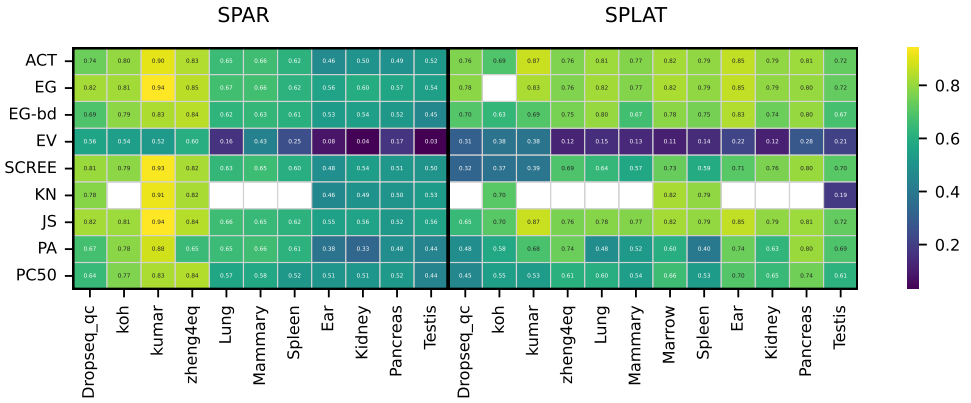


Fig. D22: Heatmap of the trustworthiness using 15 neighbors on each simulated scRNA-seq dataset where there exist a ground truth for each method. Data is standardized prior to PCA and we use 5000 highly variable genes.

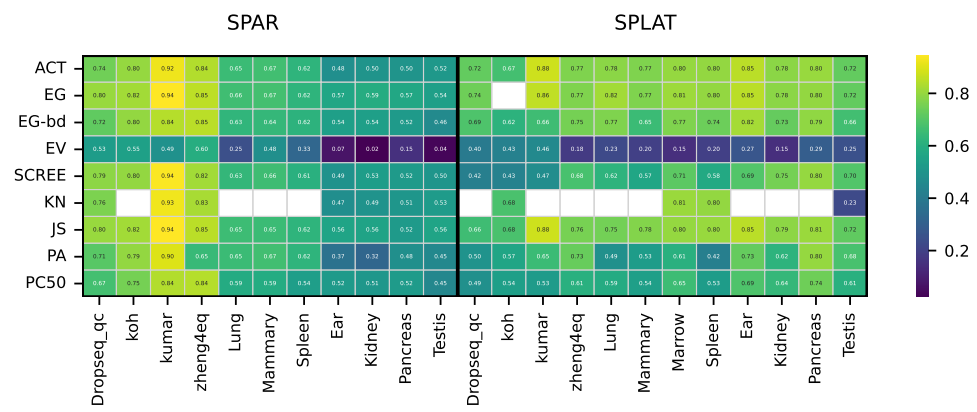


Fig. D23: Heatmap of the trustworthiness using 50 neighbors on each simulated scRNA-seq dataset where there exist a ground truth for each method. Data is standardized prior to PCA and we use 5000 highly variable genes.

Non-standardized data, and 2000 HVG

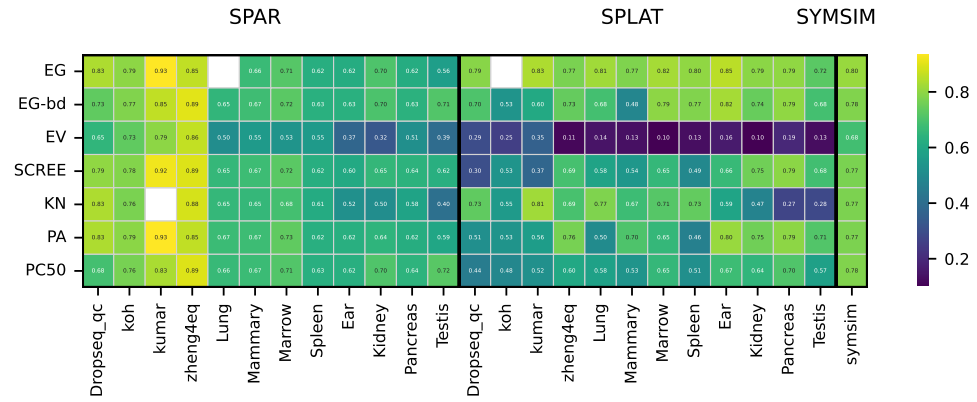


Fig. D24: Heatmap of the trustworthiness using 15 neighbors on each simulated scRNA-seq dataset where there exist a ground truth for each method. Data is not standardized prior to PCA and we use 2000 highly variable genes.

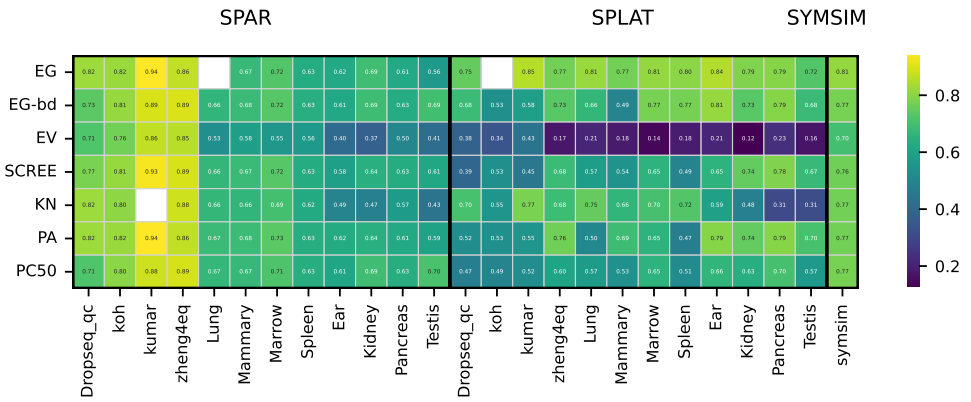


Fig. D25: Heatmap of the trustworthiness using 50 neighbors on each simulated scRNA-seq dataset where there exist a ground truth for each method. Data is not standardized prior to PCA and we use 2000 highly variable genes.

D.5 Practical summary

Table D6: Summary of the computational methods to estimate the number of components of PCA. These results are based 2000 highly variable genes and no standardization prior to PCA. Y=Yes.

		ACT	EG	EG-bd	EV	JS	KN	PA	PC50	SCREE
Real	Clustering accuracy		Tier 3	Tier 2	Tier 2		Tier 2	Tier 2	Tier 2	Tier 2
	Clustering stability		Tier 3	Tier 2	Tier 2		Tier 2	Tier 2	Tier 2	Tier 2
	Number of components		Tier 1	Tier 2	Tier 3		Tier 2	Tier 1	Tier 2	Tier 2
Simulated	Clustering accuracy		Tier 2	Tier 1	Tier 2		Tier 1	Tier 1	Tier 1	Tier 1
	Clustering stability		Tier 2	Tier 1	Tier 2		Tier 1	Tier 2	Tier 1	Tier 2
	Number of components		Tier 1	Tier 2	Tier 3		Tier 2	Tier 1	Tier 2	Tier 2
	Neigh. preserving (k=15)		Tier 1	Tier 1	Tier 3		Tier 1	Tier 1	Tier 1	Tier 1
	Neigh. preserving (k=50)		Tier 1	Tier 1	Tier 2		Tier 1	Tier 1	Tier 1	Tier 1
	Convergence success rate		Tier 3	Tier 1	Tier 1		Tier 3	Tier 1	Tier 1	Tier 1
	Scalability		Tier 1	Tier 1	Tier 1		Tier 1	Tier 1	Tier 1	Tier 1
	Speed		Tier 1	Tier 1	Tier 1		Tier 2	Tier 3	Tier 1	Tier 1
	Distributional assump.						Y			
	Require scaled data	Y				(Y)				
	Dimension constraints			Y					Y	

Tier 1 Tier 2 Tier 3 Not assessable

Table D7: Summary of the computational methods to estimate the number of components of PCA. These results are based 5000 highly variable genes and standardization prior to PCA. Y=Yes.

		ACT	EG	EG-bd	EV	JS	KN	PA	PC50	SCREE
Real	Clustering accuracy									
	Clustering stability									
	Number of components									
Simulated	Clustering accuracy									
	Clustering stability									
	Number of components									
	Neigh. preserving (k=15)									
	Neigh. preserving (k=50)									
	Convergence success rate									
	Scalability									
	Speed									
	Distributional assump.						Y			
	Require scaled data	Y				(Y)				
	Dimension constraints			Y					Y	

Tier 1 Tier 2 Tier 3 Not assessable

References

[1] Lähnemann D, Köster J, Szczurek E, McCarthy DJ, Hicks SC, Robinson MD, et al. Eleven grand challenges in single-cell data science. Genome Biology 2020 21:1. 2020 2;21(1):31–. https://doi.org/10.1186/S13059-020-1926-6.

[2] Abdellah Z, Ahmadi A, Ahmed S, Aimable M, Ainscough R, Almeida J, et al. Finishing the euchromatic sequence of the human genome. Nature 2004 431:7011. 2004 10;431(7011):931–945. https://doi.org/10.1038/nature03001.

[3] Sun S, Zhu J, Ma Y, Zhou X. Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. Genome Biology 2019 20:1. 2019 12;20(1):269–. https://doi.org/10.1186/s13059-019-1898-6.

[4] I T Jolliffe. Principal Component Analysis. 2nd ed. Springer Series in Statistics. New York: Springer-Verlag; 2002. Available from: http://link.springer.com/10.1007/b98835.

[5] Satija R, Farrell JA, Gennert D, Schier AF, Regev A. Spatial reconstruction of single-cell gene expression data. Nature Biotechnology 2015 33:5. 2015 4;33(5):495–502. https://doi.org/10.1038/nbt.3192.

[6] Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biology 2018 19:1. 2018 2;19(1):15–. https://doi.org/10.1186/s13059-017-1382-0.

[7] Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports 2019 9:1. 2019 3;9(1):5233–. https://doi.org/10.1038/s41598-019-41695-z.

[8] McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv. 2020 9;.

[9] Trapnell C, Cacchiarelli D, Grimsby J, Pokharel P, Li S, Morse M, et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nature Biotechnology 2014 32:4. 2014 3;32(4):381–386. https://doi.org/10.1038/nbt.2859.

[10] Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008 10;2008(10):P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008.

[11] Schaar A, Heumos L. Dimensionality Reduction — Single-cell best practices; 2023. Available from: https://www.sc-best-practices.org/preprocessing_visualization/dimensionality_reduction.html.

[12] Amezquita R, Lun A, Hicks S, Gottardo R. Chapter 4 Dimensionality reduction; 2021. Available from: https://bioconductor.org/books/3.23/OSCA.basic/dimensionality-reduction.html#choosing-the-number-of-pcs.

[13] Hayakawa Y, Ozaki H. A practical guide for single-cell transcriptome data analysis in neuroscience. Neuroscience Research. 2025 5;214(11):9–15. https://doi.org/10.1016/j.neures.2025.03.006.

[14] Nasrollahi FSF, Silva FN, Liu S, Chaudhuri S, Yu M, Wang J, et al. Cell Type Differentiation Using Network Clustering Algorithms. bioRxiv. 2024 12;p. 2024.12.04.626793. https://doi.org/10.1101/2024.12.04.626793.

[15] Staff TPCB. Correction: The specious art of single-cell genomics. PLOS Computational Biology. 2025 11;21(11):e1013655. https://doi.org/10.1371/JOURNAL.PCBI.1013655.

[16] Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Molecular Systems Biology 2019 15:6. 2019 6;15(6):MSB188746–. https://doi.org/10.15252/msb.20188746.

[17] Andrews TS, Kiselev VY, McCarthy D, Hemberg M. Tutorial: guidelines for the computational analysis of single-cell RNA sequencing data. Nature Protocols 2020 16:1. 2020 12;16(1):1–9. https://doi.org/10.1038/s41596-020-00409-w.

[18] Lun A. Choosing the number of principal components for denoising single-cell RNA sequencing data. University of Cambridge; 2018. Available from: https://content.cruk.cam.ac.uk/jmlab/gitlab/miscellaneous/technical-reports/pc-selection.pdf.

[19] Sciaraffa N, Gagliano A, Augugliaro L, Coronnello C. Optimization of clustering parameters for single-cell RNA analysis using intrinsic goodness metrics. Frontiers in Bioinformatics. 2025 6;5:1562410. https://doi.org/10.3389/fbinf.2025.1562410.

[20] Dobriban E, Owen AB. Deterministic Parallel Analysis: An Improved Method for Selecting Factors and Principal Components. Journal of the Royal Statistical Society Series B: Statistical Methodology. 2019 2;81(1):163–183. https://doi.org/10.1111/rssb.12301.

[21] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning. Springer Series in Statistics. New York, NY: Springer New York; 2009. Available from: http://link.springer.com/10.1007/978-0-387-84858-7.

[22] Cattell RB. The scree test for the number of factors. Multivariate Behavioral Research. 1966 4;1(2):245–276.

[23] Zhuang H, Wang H, Ji Z. findPC: An R package to automatically select the number of principal components in single-cell analysis. Bioinformatics. 2022

5;38(10):2949–2951. https://doi.org/10.1093/bioinformatics/btac235.

[24] Onatski A. Determining the number of factors from empirical distribution of eigenvalues. Review of Economics and Statistics. 2010 11;92(4):1004–1016.

[25] Linderman GC, Zhao J, Roulis M, Bielecki P, Flavell RA, Nadler B, et al. Zero-preserving imputation of single-cell RNA-seq data. Nature Communications 2022 13:1. 2022 1;13(1):192–. https://doi.org/10.1038/s41467-021-27729-z.

[26] Lun A. Using scran to analyze single-cell RNA-seq data. Bioconductor; 2019. Available from: https://bioconductor.org/packages/devel/bioc/vignettes/scran/inst/doc/scran.html#ref-lun2016step.

[27] Lun ATL, McCarthy DJ, Marioni JC. A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor. F1000Research 2016 5:2122. 2016 10;5:2122. https://doi.org/10.12688/f1000research.9501.2.

[28] Zwick WR, Velicer WF. Comparison of Five Rules for Determining the Number of Components to Retain. Psychological Bulletin. 1986 5;99(3):432–442. https://doi.org/10.1037/0033-2909.99.3.432.

[29] Fan J, Guo J, Zheng S. Estimating Number of Factors by Adjusted Eigenvalues Thresholding. Journal of the American Statistical Association. 2022;117(538):852–861. https://doi.org/10.1080/01621459.2020.1825448.

[30] Kritchman S, Nadler B. Determining the number of components in a factor model from limited noisy data. Chemometrics and Intelligent Laboratory Systems. 2008 11;94(1):19–32. https://doi.org/10.1016/j.chemolab.2008.06.002.

[31] Johnstone IM. On the distribution of the largest eigenvalue in principal components analysis. Annals of Statistics. 2001 4;29(2):295–327. https://doi.org/10.

1214/aos/1009210544.

[32] Horn JL. A rationale and test for the number of factors in factor analysis. Psychometrika. 1965 6;30(2):179–185. https://doi.org/10.1007/BF02289447.

[33] Buja A, Eyuboglu N. Remarks on Parallel Analysis. Multivariate Behavioral Research. 1992 10;27(4):509–540.

[34] Carangelo G, Magi A, Semeraro R. From multitude to singularity: An up-to-date overview of scRNA-seq data generation and analysis. Frontiers in Genetics. 2022 10;13:994069. https://doi.org/10.3389/fgene.2022.994069.

[35] Su M, Pan T, Chen QZ, Zhou WW, Gong Y, Xu G, et al. Data analysis guidelines for single-cell RNA-seq in biomedical studies and clinical applications. Military Medical Research 2022 9:1. 2022 12;9(1):68–. https://doi.org/10.1186/s40779-022-00434-8.

[36] Macosko EZ, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell. 2015 5;161(5):1202–1214. https://doi.org/10.1016/j.cell.2015.05.002.

[37] Chung NC, Storey JD. Statistical significance of variables driving systematic variation in high-dimensional data. Bioinformatics. 2015 2;31(4):545–554. https://doi.org/10.1093/bioinformatics/btu674.

[38] Satija Lab and Collaborators. Seurat - Guided Clustering Tutorial • Seurat; 2023. Available from: https://satijalab.org/seurat/articles/pbmc3k_tutorial.html.

[39] Schaum N, Karkanias J, Neff NF, May AP, Quake SR, Wyss-Coray T, et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris.

Nature 2018 562:7727. 2018 10;562(7727):367–372. https://doi.org/10.1038/s41586-018-0590-4.

[40] Quake SR. Tabula Sapiens reveals transcription factor expression, senescence effects, and sex-specific features in cell types from 28 human organs and tissues. bioRxiv : the preprint server for biology. 2025 12;https://doi.org/10.1101/2024.12.03.626516.

[41] Duò A, Robinson MD, Soneson C. A systematic performance evaluation of clustering methods for single-cell RNA-seq data. F1000Research 2020 7:1141. 2020 11;7:1141. https://doi.org/10.12688/f1000research.15666.3.

[42] Pavel A, Grønberg MG, Clemmensen LH. The impact of dropouts in scRNAseq dense neighborhood analysis. Computational and Structural Biotechnology Journal. 2025 1;27(D1):1278–1285. https://doi.org/10.1016/j.csbj.2025.03.033.

[43] Cao Y, Yang P, Yang JYH. A benchmark study of simulation methods for single-cell RNA sequencing data. Nature Communications 2021 12:1. 2021 11;12(1):6911–. https://doi.org/10.1038/s41467-021-27130-w.

[44] Baruzzo G, Patuzzi I, Di Camillo B. SPARSim single cell: a count data simulator for scRNA-seq data. Bioinformatics. 2020 3;36(5):1468–1475. https://doi.org/10.1093/bioinformatics/btz752.

[45] Sun T, Song D, Li WV, Li JJ. scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured. Genome Biology 2021 22:1. 2021 5;22(1):163–. https://doi.org/10.1186/s13059-021-02367-2.

[46] Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. Genome Biology 2017 18:1. 2017 9;18(1):174–. https://doi.org/10.1186/

s13059-017-1305-0.

[47] Zhang X, Xu C, Yosef N. Simulating multiple faceted variability in single cell RNA sequencing. Nature Communications 2019 10:1. 2019 6;10(1):2611–. https://doi.org/10.1038/s41467-019-10500-w.

[48] Rousseeuw PJ. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics. 1987 11;20(C):53–65. https://doi.org/10.1016/0377-0427(87)90125-7.

[49] Rautenstrauch P, Ohler U. Shortcomings of silhouette in single-cell integration benchmarking. Nature Biotechnology 2025. 2025 7;p. 1–5. https://doi.org/10.1038/s41587-025-02743-4.

[50] Chu LF, Leng N, Zhang J, Hou Z, Mamott D, Vereide DT, et al. Single-cell RNA-seq reveals novel regulators of human embryonic stem cell differentiation to definitive endoderm. Genome Biology 2016 17:1. 2016 8;17(1):173–. https://doi.org/10.1186/s13059-016-1033-x.

[51] Koh PW, Sinha R, Barkal AA, Morganti RM, Chen A, Weissman IL, et al. Data Descriptor: An atlas of transcriptional, chromatin accessibility, and surface marker changes in human mesoderm development Background & Summary. Scientific Data. 2016 12;3(1):160109–. https://doi.org/10.1038/sdata.2016.109.

[52] Kumar RM, Cahan P, Shalek AK, Satija R, Daleykeyser A, Li H, et al. Deconstructing transcriptional heterogeneity in pluripotent stem cells. Nature. 2014 12;516(729):56–61. https://doi.org/10.1038/nature13920.

[53] Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nature Communications 2017 8:1. 2017 1;8(1):14049–. https://doi.org/10.1038/ncomms14049.

[54] Tian L, Dong X, Freytag S, Lê Cao KA, Su S, JalalAbadi A, et al. Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments. Nature Methods 2019 16:6. 2019 5;16(6):479–487. https://doi.org/10.1038/s41592-019-0425-8.

[55] Pedregosa F, Michel V, Grisel O, Blondel M, Prettenhofer P, Weiss R, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.

[56] Hubert L, Arabie P. Comparing partitions. Journal of Classification. 1985 12;2(1):193–218. https://doi.org/10.1007/BF01908075.