bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

METHOD

Benchmarking principal component analysis for large-scale single-cell RNA-sequencing

Koki Tsuyuzaki[1], Hiroyuki Sato[2], Kenta Sato[1,3] and Itoshi Nikaido[1,4*]

*Correspondence:
itoshi.nikaido@riken.jp
[1]Laboratory for Bioinformatics Research, RIKEN Center for Biosystems Dynamics Research, Japan
Full list of author information is available at the end of the article
[†]Equal contributor

Abstract

Principal component analysis (PCA) is an essential method for analyzing single-cell RNA-seq (scRNA-seq) dataset but for large-scale scRNA-seq datasets, the computation consumes a long time and large memory space.

In this work, we review the existing fast and memory-efficient PCA algorithms and implementations and evaluate their practical application to large-scale scRNA-seq dataset. Our benchmark showed that some PCA algorithms based on Krylov subspace and randomized singular value decomposition are fast, memory-efficient, and accurate than the other algorithms. Considering the difference of computational environment of users and developers, we also developed the guideline to select the appropriate PCA implementations.

Keywords: Single-cell RNA-seq; Cellular heterogeneity; Dimension reduction; Principal component analysis; Online/Incremental algorithm; Out-of-core; R; Python; Julia

Background

Owing to the emergence of single-cell RNA sequencing (scRNA-seq) technologies [1], many types of cellular heterogeneity have been examined. For example, cellular subpopulations consisting of tissues [2–6], rare cells and stem cell niches [7], continuous gene expression change related to cell cycle [8], spatial coordinates [9–11], and difference of differentiation maturity [12, 13] have been captured by many scRNA-seq studies. Since the measurement of cellular heterogeneity highly depends on the number of cells measured simultaneously, a wide variety of large-scale scRNA-seq technologies have been developed [14], including those using cell sorting devices [15–17], Fludigm C1 [18–21], droplet-based technologies (Drop-Seq [2–4], in-Drop RNA-Seq [5, 6], 10X Genomics Chromium [22]), and single-cell combinatorial-indexing RNA-sequencing (sci-RNA-seq [23]). Such technologies have encouraged the establishment of several large-scale genomics consortiums such as the Human Cell Atlas [24–26], Mouse Cell Atlas [27], and Tabula Muris [28]. These projects are measuring a tremendous number of cells by scRNA-seq and tackling basic life science problems such as the number of cell types consisting of an individual, cell-type-specific marker gene expression and gene functions, and molecular mechanisms of diseases at a single-cell resolution.

Nevertheless, the analysis of scRNA-seq datasets poses a potentially difficult problem; the cell type corresponding to each data point is unknown *a priori* [1, 29–33]. Accordingly, researchers perform unsupervised machine learning (UML) methods such as dimensional reduction and clustering to reveal the cell type corresponding to each individual data point. In particular, principal component analysis

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

(PCA [34–36]) is a workhorse algorithm for UML across many situations. PCA is widely used for data visualization [37–39], data quality control (QC) [40], feature selection [13,41–47], de-noising [48,49], imputation [50–52], confirmation and removal of batch effects [53–55], confirmation and estimation of cell-cycle effects [56], input of other non-linear dimensional reduction [57–63] and clustering methods [64–67], rare cell type detection [68,69], cell type and cell state similarity search [70], pseudotime coordinate [13, 71–75], and spatial coordinate [9]. A wide variety of data analysis pipelines include PCA as an internal function or utilize principal component (PC) scores as input for the down-stream analyses [22, 76–83].

Despite its wide use, there are several reasons why it is unclear how PCA should be conducted for large-scale scRNA-seq. First, since the widely used PCA algorithms and implementations load all elements of data matrix into memory space, for large-scale datasets such as the 1.3 million cells measured by 10X Genomics Chromium [39] or the 2.0 million cells measured by sci-RNA-seq [23], the calculation is difficult unless the memory size of the user's machine is very large. Furthermore, the same data analysis workflow is repeatedly performed, with deletions or additions of the data or changes of parameters of the workflow, and under such trial-and-error cycles, PCA can become a bottleneck of the workflow. Therefore, some fast and memory-efficient PCA algorithms are required.

Second, there are indeed other PCA algorithms that are faster and more memory-efficient, but their practicality for use with large-scale scRNA-seq datasets is not fully known. Generally, the acceleration of algorithms by some approximation methods, and the accuracy of biological data analysis can be a trade-off. Fast PCA algorithms might overlook some important differential gene expression. In the case of large-scale scRNA-seq studies aiming to find novel cell types, this property may cause the loss of clustering accuracy and not acceptable.

Finally, actual computational time and memory efficiency are highly dependent on the specific implementation, including the programming language and data format, but there is no benchmarking for evaluating these properties. Such information is directly related to the practicality of the software and is useful as a guideline for users and developers.

For the above reasons, in this work, we examine the practicality of fast and memory-efficient PCA algorithms for use with large-scale scRNA-seq datasets. This work provides four key contributions. First, we reviewed the existing PCA algorithms and their implementations (Figure 1). Second, we performed a benchmark test with selected PCA algorithms and implementations. To our knowledge, this is the first comprehensive benchmarking of PCA with scRNA-seq datasets. Third, we provide some original implementations of some PCA algorithms and utility functions for QC, filtering, and feature selection. All commands are implemented as a fast and memory-efficient Julia package. Finally, we propose guidelines for end-users and software developers.

Results

Review of PCA algorithms and implementations

Here, we review the existing PCA algorithms and their implementations. All algorithms pseudo-code is provided in Additional file 1.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

PCA is formalized as eigenvalue decomposition (EVD) of the covariance matrix of the data matrix or singular value decomposition (SVD) of the data matrix [84]. To perform PCA, the most widely used PCA function in the R language is probably `prcomp` function, which is a standard R function [13, 40, 41, 49, 53, 64, 67, 69, 72, 74, 76, 79, 85]. Likewise, users and developers of the Python language may use the `PCA` function of *scikit-learn* (*sklearn*) [50, 54, 56, 86, 87], which is a Python package for machine learning. These are actually wrapper functions for performing SVD with LAPACK subroutines such as `DGESVD` (QR method-based) or `DGESDD` (divide-and-conquer method-based), and both subroutines perform the Golub-Kahan method [84]. In this method, the covariance matrix is tri-diagonalized by Householder transformation, and then the tri-diagonalized matrix is diagonalized by the QR method or divide-and-conquer method (Figure 2a). Such a two-step transformation is commonly performed by the sequential similarity transformation expressed as $M_k^{-1}...M_2^{-1}M_1^{-1}XM_1M_2...M_k$, where X is an n-by-n covariance matrix, M_k is an n-by-n invertible matrix, and k is the step of the transformation. Likewise, when the input data matrix is asymmetric, the matrix is bi-diagonalized and then tri-diagonalized. When the matrix is finally diagonalized at the kth step, the diagonal elements become eigenvalues and $M = M_1M_2...M_k$ becomes the set of corresponding eigenvectors. Although the Golub-Kahan method is the most widely used SVD algorithm, this method has some drawbacks. First, the large dense matrix M must be temporarily saved and incrementally updated in each step, and therefore the memory space is filled quickly. Second, when the matrix is large, the data matrix itself is difficult to be loaded and causes an out-of-memory error. For the above reasons, the Golub-Kahan method cannot be directly applied to large-scale scRNA-seq datasets.

There are some faster and more memory-efficient PCA algorithms. Contrary to the full-rank SVD solved by LAPACK, such algorithms are formalized as truncated SVD, in which only some of the top PCs are calculated. We classify these methods into five categories (Figure 2b). The first category consists of downsampling-based methods [88]. In these methods, SVD is first performed for a small matrix consisting of cells randomly sampled from the original large matrix. The remaining cells are then projected onto the low-dimensional space spanned by the eigenvectors learned from the small matrix. The effectiveness of this method in scRNA-seq studies has been evaluated by Bhaduri *et al.* [88] (Table 1).

The second category is SVD update [89], which repeatedly performs SVD using subsets of the data sampled from the data matrix and incrementally updates the result. Sequential Karhunen-Loeve transform (SKL) [89], which is a kind of SVD update, is used in *loompy* (`http://loompy.org`) (Table 1).

The third category consists of Krylov subspace-based methods [90–93]. The most typical method within this category is the power method, which iteratively multiplies a vector w with a covariance matrix X and normalizes w. Within some iterations, w converges to the eigenvector (PC1) corresponding to the largest eigenvalue. Since the way to calculate the higher PCs is not obvious, there are some algorithms, such as orthogonal iteration (block power method, subspace iteration, or simultaneous iteration [90]), the Lanczos method [90], and the Arnoldi method [90]. Orthogonal iteration performs the power method with multiple initial vectors in parallel,

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

and each power iteration step performs QR decomposition for orthonormalization. Contrary to orthogonal iteration, Lanczos and Arnoldi methods respectively introduce Lanczos and Arnoldi processes to generate vectors that are orthogonal to each other. To make the convergence faster, both methods also introduce "restart" strategies such as the augmented implicitly restarted Lanczos bidiagonalization algorithm (IRLBA [94]) and implicitly restarted Arnoldi methods (IRAM [95]), in which new initial vectors are calculated by the accumulated result of Lanczos or Arnoldi processes. In contrast to the Golub-Kahan method, these methods do not generate large dense temporary matrices, and when the data matrix is sparse, these methods are compatible with a sparse matrix format and can be accelerated. *Cell Ranger* [22], *Seurat2* [47], *Scanpy* [87], *SAFE* [66], *Scran* [48], *Giniclust2* [68], *MAGIC* [50], *Harmony* [55], and *Scater* [76] use IRLBA for PCA functions (Table 1). Although IRAM appears not to have been used in scRNA-seq studies, the effectiveness of its use with population genetic datasets has been argued recently [96].

The fourth category is gradient descent (GD, or steepest descent)-based methods. In this method category, the gradient of the objective function is calculated, and the initial vectors are updated to the reverse direction of the gradient. Although GD utilizes all the data to calculate the gradient (i.e., full gradient), stochastic gradient descent (SGD) calculates the gradient with a subset of the data (i.e., stochastic gradient). Although these PCA algorithms are sometimes used for situations in which the data are incrementally observed, such as subspace tracking [97], these methods also can be fast and memory-efficient because the calculation of the full/stochastic gradient is decomposable to the sum of the gradient of individual data points. Although these methods appear not to have been used in scRNA-seq studies, in image processing studies, this method is known as Oja's method or the generalized Hebbian algorithm [98–100].

The fifth category comprises random projection-based methods, in which a data matrix is randomly projected onto lower dimensions and basic linear algebraic methods such as QR decomposition and SVD are performed on the smaller matrix. Since most calculations are performed for these random lower dimensions, these methods can be fast and memory-efficient. Surprisingly, in this method, the SVD of the original data matrix can be accurately reconstructed from the arithmetic of the small matrix with low reconstruction error [101,102]. Although Halko's method is known as an algorithm of the randomized SVD, Li *et al.* also modified the preconditioning step so that the calculation time is improved (algorithm971 [103]). Halko's method is used in *scanpy* [87], *SIMLR* [65], and *SEQC* [83], and algorithm971 is implemented in *CellFishing.jl* [70] (Table 1). Halko's method is also used in population genetic studies [104].

Notably, the acceleration techniques of the above algorithms are based on random row/column selection or random projection of data matrices, both of which are used to make a large matrix smaller. When these processes are used in an out-of-core (also known as, online, incremental, or on-disk) implementation, in which only a subset of the data matrix is loaded into the memory and used to incrementally update the calculation, these algorithms might be scalable to even scRNA-seq datasets consisting of millions of cells. For example, in fast Fourier transform-accelerated interpolation-based t-stochastic neighbor embedding (*FIt-SNE* [59]), algorithm971

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

is implemented in an out-of-core manner and named out-of-core PCA (oocPCA). However, most PCA implementations load all the elements of a data matrix into the memory-space simultaneously. Therefore, the order of memory usage of such algorithms is commonly $\mathcal{O}(NM)$, where N is the number of genes and M is the number of cells (Figure 3). To extend the scope of algorithms used in the benchmarking, we originally implemented algorithms such as orthogonal iteration, GD, SGD, Halko's method, and algorithm971 in an out-of-core manner.

Benchmarking of PCA algorithms and implementations

Here, we perform the benchmarking test of the PCA algorithms described above. We list PCA implementations that are freely available, easily downloaded, installed, and performed as well as possible. The source code for performing the benchmarking is summarized in Additional file 2, and the results of the benchmarking are summarized in Figure 3.

Real-world datasets

In consideration of the trade-offs among the large number of methods to be evaluated and our limited time, computational resources, and manpower, we carefully selected real-world datasets for benchmarking. We selected three datasets: mouse cells from a primary visual cortex region (Cortex), human cells from the pancreas (Pancreas), and mouse cells from the cortex, hippocampus, and ventricular zone (Brain) (Table 2). These datasets have been used in many previous scRNA-seq studies [66, 70, 82, 88, 105–111].

The accuracy of PCA algorithms

Here, we evaluate the accuracy of the various PCA algorithms by using three real-world datasets. For the analyses of the Cortex and Pancreas datasets, we set the result of `prcomp` as the gold standard, and the other implementations are compared with this result (Figure 1b and 3). For the Brain dataset analyses, full-rank SVD by LAPACK is computationally difficult. Therefore, we set the result of *Cell Ranger* as the gold standard. Although we know that the algorithm is IRLBA, some details of data preprocessing, such as gene selection and logarithm transformation, are unclear. Accordingly, for the Brain dataset, the comparison may be imprecise. In our computing environment, we could not use *Cell Ranger* to analyze the Brain dataset owing to an out-of-memory error in Python, so the result of the analysis provided by 10X Genomics is used instead.

First, we performed t-stochastic neighbor embedding (t-SNE [57, 58]) for the results of each PCA algorithm and compared the clarity of the cluster structure detected by the original studies (Figure 1b and 4). For the Brain dataset, only downsampling and some out-of-core PCA implementations such as `IncrementalPCA` (*sklearn*), `orthiter`/`gd`/`sgd`/`halko`/`algorithm971` (*OnlinePCA.jl*), and `oocPCA_CSV` (*oocRPCA*) could be performed, while the other implementations were terminated by out-of-memory errors. Compared with the gold standard cluster structures, the structures detected by downsampling were unclear, and some distinct clusters were incorrectly combined into single clusters. In the realistic situation when the cellular labels are not available *a priori*, the labels were exploratorily estimated by confirming differentially expressed genes, known marker-genes, or related gene functions

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

of clusters. In such a situation, downsampling may overlook subgroups hiding in a cluster. We also performed two clustering methods (k-means and Gaussian mixture model (GMM) clustering [112]) against all the results of the PCA implementations and calculated the adjusted Rand index (ARI [113]) to evaluate clustering accuracy (Figure 1b and 5). Compared with the gold standard, the result of downsampling and `sgd` (*OnlinePCA.jl*) were worse, and the other implementations were as accurate as the gold standard.

Next, we performed an all-to-all comparison between PCs from the gold standard and the other PCA implementations (Figure 1b and 6). Since the PCs are unit vectors, when two PCs are directed in the same or opposite direction, their cross product becomes 1 or -1, respectively. Both the same and opposite direction vectors are mathematically identical in PCA optimization problem and different PCA implementations may yield PCs with different signs. Accordingly, we calculated the absolute value of the cross product ranging from 0 to 1 for the all-to-all comparison and evaluated whether higher PCs are accurately calculated. The figure 6 shows that the higher PCs of downsampling, `orthiter`/`gd`/`sgd` (*OnlinePCA.jl*), and `PCA` (*dask-ml* [114]) become inaccurate as the dimensionality of the PC increases. The higher PCs of these implementations also look noisy and unclear in pair plots of PCs in each implementation and seem uninformative (Additional file 3, Additional file 4, and Additional file 5). In particular, the higher PCs calculated by `sgd` (*OnlinePCA.jl*) are sometimes influenced by the existence of outlier cells (Additional file 3, Additional file 4, and Additional file 5) and very sensitive to the different learning parameters, the number of row vectors in the data matrix (i.e., number of epoch or pass out, Additional file 7). Contrary to these results, all the implementations of IRLBA and IRAM as well as the randomized SVD approaches except for `PCA` (*dask-ml*) are surprisingly accurate regardless of the difference in the written language and the developers. Although `PCA` (*dask-ml*) is based on Halko's method and almost identical to the other implementations of Halko's method, this function uses the direct tall-and-skinny QR algorithm [115] (`https://github.com/dask/dask/blob/a7bf545580c5cd4180373b5a2774276c2ccbb573/dask/array/linalg.py#L52`) and this part might be related to the inaccuracy.

For the Brain dataset, compared with the gold standard (`irlb` (*Cell Ranger*)), the diagonal lines within the plots of all the results seem unclear (Figure 6c). This may be because the data preprocessing condition for `irlb` (*Cell Ranger*) [22] and the other PCA implementations are not identical. The distribution of eigenvalues of `irlb` (*Cell Ranger*) is also slightly flat compared with the other implementations (Figure 7c).

Because PCA calculates cell-wise eigenvectors (PCs) and gene-wise eigenvectors (loading vectors) simultaneously, we also performed all-to-all comparisons between the loading vectors of the gold standard and those of the other PCA implementations (Figure 8). We extracted the top 500 genes in terms of the largest absolute values in loading vectors and calculated the number of genes in common between the two loading vectors. The same tendencies were observed even in loading vectors. Since the genes with large absolute values in loading vectors are used as feature values in some studies [41–46], inaccurate PCA implementations may lower the accuracy of such an approach. The distribution of the eigenvalues of downsampling,

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

`IncrementalPCA` (*sklearn*), and `sgd` (*OnlinePCA.jl*) are also different from those of the other implementations (Figure 7).

Finally, we compared the computational time and the memory usage of all the PCA implementations (Figure 9). For the Brain dataset, downsampling itself was fast, but the preprocessing steps, such as matrix transposition (X') and multiplication of the transposed data matrix and the loading vectors to calculate PCs ($X'W$), were slow and had high memory space requirements (Additional file 2). We also found that the calculation time of `PCA` (*dask-ml*) was not as fast in spite of its out-of-core implementation; for the Brain dataset, this implementation could not finish the calculation within three days in our computational environment. The other out-of-core PCA implementations such as `IncrementalPCA` (*sklearn*), `orthiter/gd/sgd/halko/algorithm971` (*OnlinePCA.jl*), and `oocPCA_CSV` (*oocRPCA*), were able to finish those calculations in 10 or fewer hours.

Calculation time, memory usage, and scalability

We also systemically estimated the calculation time, memory usage, and scalability of all the PCA implementations using 18 synthetic datasets consisting of $\{10^2,10^3,10^4\}$ genes $\times$ $\{10^2,10^3,10^4,10^5,10^6,10^7\}$ cells matrices (see Materials and methods). We evaluated whether the calculations can be finished or are terminated by out-of-memory errors (Figure 1b). We also manually terminated a PCA process (i.e., *dask-ml*) that was unable to generate output files within three days. All the terminated jobs are summarized in Additional file 6. Note that the number of epochs in `orthiter/gd/sgd` (*OnlinePCA.jl*) is one, and in most situations, the value should be tuned using grid search (Additional file 7).

Figures 10 and 11 show the calculation time and the memory usage of all the PCA implementations, which can be scaled to a $10^4 \times 10^7$ matrix. `IncrementalPCA` (*sklearn*) and `oocPCA_CSV` (*oocRPCA*) were slightly slower than the other implementations (Figure 10), and this was probably because the inputs of these implementations were CSV files while the other implementations used binary files. The memory usage of all the implementations were almost the same except for `oocPCA_CSV` (*oocRPCA*). This is probably because this function has a parameter that controls the maximum memory usage (*mem*), and we set the value as 10 GB (Additional file 2). Indeed, the memory usage seemed to have converged to around 10 GB (Figure 11). This property is considered an advantage of this implementation; the users can specify different values to suit the computational environment.

The relationship between file format and performance

We also counted the pass out of the Brain matrix in out-of-core PCA implementations (Figure 12a) and found that the calculation time was correlated with the number of pass out of the implementation. Furthermore, data compression substantially accelerates the calculation time. This suggests that the data loading process is very critical for out-of-core implementation and that the overhead for this process has a great effect on the overall calculation time and memory usage. Accordingly, using different data formats, such as CSV, Zstd, Loom [87], and hierarchical data format 5 (HDF5), provided by the 10X Genomics (10X-HDF5) of the Brain dataset, we evaluated the calculation time and the memory usage for the simple one-pass

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

orthogonal iteration (qr(XW)), where qr is the QR decomposition, X is the data matrix, and W is the 30 vectors to be estimated as the eigenvectors (Figure 12b). Since only one row is loaded at once for CSV, Zstd, and Loom formats, their memory usage is very low but the time needed for calculation is greater than that of 10X-HDF5. Conversely, for 10X-HDF5 format, the data matrix is stored as compressed sparse column format (CSC), which omits the 0 values and saves memory usage, and this enables the large data matrix to be divided into multiple blocks containing multiple row vectors, for which each block is loaded incrementally. While it is not obvious that the usage of sparse matrix accelerates the PCA with scRNA-seq datasets because scRNA-seq datasets are not particularly sparse compared with data from other scientific fields (cf. recommender systems or social network [116, 117]), we showed that it does speed up the calculation time for scRNA-seq datasets.

When all row vectors stored in 10X-HDF5 are loaded at once, the calculation is fastest, but the memory usage is also highest. Since the calculation time and the memory usage have a trade-off and the user's computational environment is not always high-spec, the block size should be optionally specified as an argument of the command. For the above reasons, we also developed `tenxpca`, which is a new implementation that performs algorithm971 for sparse matrix stored in 10X-HDF5 format. Using all elements of the CSC at once, `tenxpca` can finish the calculation in 1.18 hours with 82.96 GB memory usage. This is the fastest analysis of the Brain dataset in this study. According to the user's machine specification, the number of rows loaded at once can be optionally changed to a different number.

In addition to `tenxpca`, some algorithms used in this benchmarking, such as orthogonal iteration, GD, SGD, Halko's method, and algorithm971, are implemented as Julia functions and command line, which have been published as a Julia package *OnlinePCA.jl* (Figure 13). When the data are stored as a CSV file, they are compressed as a Zstd file (Figure 13a) and then some out-of-core PCA implementations are performed. When the data are in 10X-HDF5 format, algorithm971 is directly performed with the data by `tenxpca` (Figure 13b). We also implemented some functions and command line to extract row-wise/column-wise statistics such as mean and variance as well as highly variable genes [118] in an out-of-core manner. Because such statistics are saved as small vectors, they can be loaded by any programming language and applied to QC, and the users can select only informative genes and cells. After QC, the filtering command removes low-quality genes/cells and generates another Zstd file.

Guidelines for users and developers

Based on all the benchmarking results and our implementation in this work, we propose some user guidelines (Figure 14). Considering that bioinformatics studies combine multiple tools to construct a user's specific workflow, written language is an important factor in selecting the right PCA implementation. Therefore, we categorized the PCA implementations by their written language (i.e., R, Python, and Julia; Figure 14, column-wise). Along with the data matrix size, we also categorized implementations by the way they load data (in-memory or out-of-core) as well as their input matrix format (dense or sparse, Figure 14, row-wise). Here we define the GC-value of a data matrix as the number of genes $\times$ the number of cells.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

If the data matrix is not too large (e.g., GC $\leq 10^7$), the data matrix can loaded as a dense matrix, and full-rank SVD of LAPACK is then accurate and optimal (in-memory & dense matrix). In such a situation, the wrapper functions for utilizing LAPACK written in each language are useful. However, if the data matrix is much larger (e.g., GC $\geq 10^8$), an alternative to LAPACK is needed. Based on the benchmarking results, we recommend IRLBA, IRAM, Halko's method, and algorithm971 as alternatives to LAPACK. If the GC-value is around $10^8 \leq$ GC $\leq 10^{10}$, and if the data matrix can be loaded into memory as a sparse matrix, some implementations for these algorithms are available (in-memory & sparse matrix). In particular, such implementations are effective for large data matrices stored in 10X-HDF5 format as CSC format. Seurat2 [47] also introduces this approach by combining the matrix market format (R, *Matrix*) and `irlba` function (R, *irlba*). When the data matrix is dense and cannot be loaded into memory space (e.g., GC $\geq 10^{10}$), the out-of-core implementations such as `oocPCA_CSV` (R, *oocRPCA*), `IncrementalPCA` (Python, *sklearn*), and `algorithm971` (Julia, *OnlinePCA.jl*) are useful (dense matrix & out-of-core). If the data matrix is extremely large and cannot be loaded into memory even if the data are formatted as a sparse matrix, out-of-core PCA implementations for sparse matrix are needed. In such a situation, `tenxpca` can be used if the data is stored in 10X-HDF5 format.

There is a point to be noted regarding effective utilization of the implementations for randomized SVD. Both Halko's method and algorithm971 have a parameter for specifying the number of power iterations (niter), and this iteration step sharpens the distribution of eigenvalues and enforces a more rapid decay of the singular values ([119] and Additional file 2). In our experiments, the value of niter is very critical for achieving accuracy, and we highly recommend niter values of 3 or larger (Additional file 8). In some implementations, this parameter is specified as a smaller number or cannot be accessed as a function parameter. Therefore, users should carefully set the parameter or select an appropriate implementation.

We also propose guidelines for developers. To develop fast, memory-efficient, and scalable PCA implementations, there are many data, algorithms, and computational framework and environment methodologies (Additional file 9). Here, we focus on two topics.

The first topic is "loss of sparsity". As described above, the usage of sparse matrix can effectively reduce the memory space and accelerate the calculation, but developers must be careful not to destroy the sparsity of a sparse matrix. PCA with a sparse matrix is not equivalent to SVD with sparse matrix; in PCA, all sparse matrix elements must be centered by the substitution of gene-wise average values. Once the sparse matrix X is centered ($X - X_{mean}$), it becomes a dense matrix filled with floating-point numbers, and the memory usage is significantly increased. Obviously, the explicit calculation should be avoided. In such a situation, if multiplication of this centered matrix and dense vector/matrix is required, the calculation should be divided into two parts, such as $(X - X_{mean})\,W = XW - X_{mean}W$, and these parts should be calculated separately. If one or both parts require more than the available memory space, such parts should be incrementally calculated in an out-of-core manner. There are actually some PCA implementations that can accept a sparse matrix, but they may consume very long calculation time and large memory space because of a loss of sparsity (cf. `rpca` of *rsvd* https://github.com/cran/rsvd/blob/

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

7a409fe77b220c26e88d29f393fe12a20a5f24fb/R/rpca.R#L158). To the best of our knowledge, `prcomp_irlba` of *irlba* (https://github.com/bwlewis/irlba/blob/8aa970a7d399b46f0d5ad90fb8a29d5991051bfe/R/irlba.R#L379), `irlb` of *Cell Ranger* (https://github.com/10XGenomics/cellranger/blob/e5396c6c444acec6af84caa7d3655dd33a162 lib/python/cellranger/analysis/irlb.py#L118), `safe_sparse_dot` of *sklearn* (https://scikit-learn.org/stable/modules/generated/sklearn.utils.extmath.safe_sparse_dot.html), and `tenxpca` of *OnlinePCA.jl* (https://github.com/rikenbit/OnlinePCA.jl/blob/c95a2455acdd9ee14f8833dc5c53615d5e24b5f1/src/tenxpca.jl#L183) deal with this topic. Likewise, as an alternative to the centering calculation, `MaxAbsScaler` of *sklearn* (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.htr introduces a scaling method, in which the maximum absolute value of each gene vector becomes one, thereby avoiding the loss of sparsity.

The second topic is "lazy loading." Although, the out-of-core PCA implementations used in this benchmarking explicitly calculate centering, scaling, and any other arithmetic operations from the extracted block of the data matrix, such processes should be virtually calculated, as if the matrix was in memory. In this situation, only when the data are actually required, the processes should be lazily calculated on the fly. Source code to achieve this is even safe and readable. Some packages, such as `DeferredMatrix` of *BiocSingular* (R/Bioconductor, https://bioconductor.org/packages/devel/bioc/html/BiocSingular.html), *CenteredSparseMatrix* (Julia, https://github.com/jsams/CenteredSparseMatrix), *Dask* [114] (Python, https://dask.org), and *Vaex* (Python, https://vaex.io/), support lazy loading.

Discussion

In this benchmarking study, we found that PCA implementations based on LAPACK are accurate but cannot be scaled for use with large-scale scRNA-seq datasets such as the Brain dataset, and alternative implementations are thus required. Some methods approximate the calculation by using truncated SVD forms such as IRLBA, IRAM, Halko's method, and algorithm971, and these are sufficiently accurate as well as faster and more memory-efficient than LAPACK. The actual memory usage is highly dependent on whether an algorithm is implemented as out-of-core or whether sparse matrix can be specified as input. Some sophisticated implementations, including ours, can handle such issues. Other PCA algorithms, such as downsampling, SKL, orthogonal iteration, GD, and SGD, are actually not accurate, and their use risks overlooking cellular subgroups contained within scRNA-seq datasets. These methods commonly update eigenvectors with small fractions of the data matrix, and this process may overlook subgroups or subgroup-related gene expression, thereby causing the observed inaccuracy. Although the down-stream analyses of PCA vary widely, and we could not examine all the topics of scRNA-seq analysis, such as rare cell-type detection [68, 69] and pseudotime analysis [13, 71–75], differences among PCA algorithms might also affect the accuracy of such analyses. Butler *et al.* showed batch effect removal can be formalized as canonical correlation analysis (CCA) [47], which is mathematically very similar to PCA. The optimization of CCA is also formalized in various ways, including randomized CCA [120] or SGD of CCA [121]. Although this topic is beyond the scope of the present work, we will also evaluate such differences among algorithms in the future.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

This work also sheds light on the effectiveness of randomized SVD. This algorithm is popular in population genetic studies [104]. In the present study, we also assessed its effectiveness with scRNA-seq datasets with high heterogeneity. This algorithm is relatively simple and some studies have implemented it from scratch (Table 1). The simplicity may be the attraction of this algorithm. Our literature review, benchmarking, special implementation for scRNA-seq datasets, and guidelines provide important resources for new users and developers tackling the challenges of large-scale scRNA-seq data analysis. EVD/SVD is also known as the "master" algorithm for matrices [84]; this method also can solve least squares problems and be applied to other data analysis methods, such as dimensional reduction, clustering, and prediction, which means many out-of-core algorithms can be developed for large scRNA-Seq datasets.

Materials and methods

Empirical datasets

The gene expression matrix and cell type labels for the Cortex dataset [37] were retrieved from the Single Cell Portal Beta (https://portals.broadinstitute.org/single_cell/study/a-transcriptomic-taxonomy-of-adult-mouse-visual-cortex-visp). The gene expression matrix and cell type labels for the Pancreas dataset [38] were retrieved from the GEO database (GSE84133). The gene expression matrix and cell type labels for the Brain dataset [39] were downloaded from the 10X Genomics company website (https://support.10xgenomics.com/single-cell/datasets/1M_neurons). The genes of all matrices with zero variance were removed because such genes are meaningless for PCA calculation. The number of remaining genes and cells are summarized in Table 2.

Simulated datasets

All count datasets were generated by the R `rnbinom` (random number based on a negative binomial distribution) function with shape and rate parameters of 0.4 and 0.3, respectively. Matrices of $\{10^2,10^3,10^4\}$ genes $\times$ $\{10^2,10^3,10^4,10^5,10^6,10^7\}$ cells were generated.

Benchmarking procedures

Assuming digital expression matrices of unique molecular identifier (UMI)-counts, all the data files, including real and synthetic datasets, were in CSV format. When using the Brain dataset, the matrix stored in 10X-HDF5 format was converted to CSV using our in-house Python script (https://gist.github.com/kokitsuyuzaki/5b6cebcaf37100c8794bdb89c7135fd5). After being loaded by each PCA implementation, the raw data matrix X_{raw} was transformed to X by the logarithm-transformation $X = \log_{10}(X_{\mathrm{raw}} + 1)$, where *log* is the element-wise logarithm. When performing each PCA implementation based on the truncated SVD, the number of PCs were specified in advance (Table 2).

Although it is unclear how many cells should be used in downsampling, an empirical analysis [88] suggests that 20,000 to 50,000 cells are sufficient for clustering and detecting subpopulations in the 1.3M dataset. Thus $50000/1300000 \times 100 = 3.8\%$ of cells were sampled from each dataset and used for the downsampling method. When

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

performing `IncrementalPCA` (*sklearn*), the row-vectors, which match the number of PCs, were extracted until the end of the lines of the files. When performing `irlb`(*Cell Ranger*), the loaded dataset was first converted to the scipy sparse matrix and specified with the function. This is because this function supports sparse matrix data stored in 10X-HDF5 format. When performing the benchmark, this conversion time and memory usage were also included. When performing all the functions of *OnlinePCA.jl* such as `orthiter`/`gd`/`sgd`/`halko`/`algorithm971`, we converted the CSV data to Zstd format, and the calculation time and the memory usage were included in the benchmark for fairness. `orthiter`, `gd`, and `sgd` (*OnlinePCA.jl*) were performed until the calculations converged (Additional file 7). For all the randomized SVD implementations, the niter parameter value was set to 3 (Additional file 8). When performing `oocPCA_CSV`, the users can also use `oocPCA_BIN`, which is used to perform PCA with binarized CSV files. The binarization is performed by the `csv2binary` function, which is also implemented in the *oocRPCA* package. Although data binarization will accelerate the calculation time for PCA itself, we confirmed that `csv2binary` is based on the in-memory calculation, and in our environment, `csv2binary` was terminated by an out-of-memory error. Accordingly, we only used `oocPCA_CSV`, and the CSV files were directly loaded by this function.

Since most algorithms are based on random numbers, we also tried to evaluate stability, as captured by variation among multiple trials. However, we could not specify the random seed in many of the implementations. This is because, in many cases, there is no parameter for specifying the seed in the PCA function, or sometimes the source code for performing the PCA calculation is separated into other languages such as FORTRAN, C, and C++ making the seed hard to specify in the code. We confirmed that many implementations generated the same results with multiple trials (data not shown), but this does not always mean the calculations stably converge to the same solution from any random seed, because the random seed is sometimes hard-coded in the source code. For the above reason, in this work, we used the result of a single trial for each implementation.

Computational environment

All computations were performed on two node-machines with Intel Xeon E5-2697 v2 (2.70 GHz) processors and 128 GB of RAM, four node-machines with Intel Xeon E5-2670 v3 (2.30 GHz) processors and 96 GB of RAM, and four node-machines with Intel Xeon E5-2680 v3 (2.50 GHz) processors and 128 GB of RAM. Storage among node machines was shared by NFS, connected using InfiniBand. All jobs were queued by the Open Grid Scheduler/Grid Engine (v2011.11) in parallel. The elapsed time and maximum memory usage were evaluated using the GNU `time` command (v1.7). We also tried to use *Cell Ranger* (v1.3.0) to analyze the Brain dataset on a large-memory machine with an Intel Xeon (2.90 GHz) processor and 512 GB of RAM.

Reproducibility

All the analyses were performed on the machines described above. We used R v3.5.0, Python v3.6.4, and Julia v1.0.1 in the benchmarking, and only when we performed t-SNE by bhtsne (`https://github.com/lvdmaaten/bhtsne`) and CSV

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

conversion of Brain dataset, we used Python v2.7.9. All the programs used to perform the PCA implementations in the benchmarking are summarized in Additional file 2. Orthogonal iteration, GD, SGD, Halko's method, and algorithm971 are implemented as `orthiter`, `gd`, `sgd`, `halko`, and `algorithm971`, respectively, which are the Julia functions or commands for *OnlinePCA.jl* (https://github.com/rikenbit/OnlinePCA.jl). We also published the script files used to perform the benchmark (https://github.com/rikenbit/onlinePCA-experiments).

Abbreviations

PCA: principal component analysis; scRNA-seq: single-cell RNA sequencing; sci-RNA-seq: single-cell combinatorial-indexing RNA-sequencing analysis; UML: unsupervised machine learning; QC: quality control; PC: principal component; EVD: eigenvalue decomposition; SVD: singular value decomposition; *Sklearn*: *scikit-learn*; SKL: sequential Karhunen-Loeve transform; IRLBA: augmented implicitly restarted Lanczos bidiagonalization; IRAM: implicitly restarted Arnoldi method; GD: gradient descent; SGD: stochastic gradient descent; t-SNE: t-stochastic neighbor embedding; FIt-SNE: fast Fourier transform-accelerated interpolation-based t-stochastic neighbor embedding; oocPCA: out-of-core PCA; GMM: Gaussian mixture model; ARI: adjusted Rand index; Zstd: Zstandard; UMI: unique molecular identifier; CSV: comma-separated values; HDF5: hierarchical data format 5; 10X-HDF5: HDF5 provided by 10X Genomics; CSC: compressed sparse column format; CSR: compressed sparse row format

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by MEXT KAKENHI Grant Number 16K16152. This work was partially supported by the Japan Science and Technology Agency (JST), CREST grant number JPMJCR16G3, and the Projects for Technological Development, Research Center Network for Realization of Regenerative Medicine by Japan (18bm0404024h0001), the Japan Agency for Medical Research and Development (AMED).

Author's contributions

KT and HS surveyed the PCA algorithms and implementations. KT and IN designed the benchmarking test. KT and KS implemented the Julia program and performed all the analyses. KT retrieved and preprocessed the test dataset to evaluate the proposed method. All the authors have written, read, and approved the manuscript.

Acknowledgements

We thank Mr. Akihiro Matsushima and Mr. Manabu Ishii for their assistance with the IT infrastructure for the data analysis. We are also grateful to all member of the Laboratory for Bioinformatics Research, RIKEN Center for Biosystems Dynamics Research for their helpful advice. Computations were partially performed on the NIG supercomputer at ROIS National Institute of Genetics.

Author details

[1]Laboratory for Bioinformatics Research, RIKEN Center for Biosystems Dynamics Research, Japan. [2]The Hakubi Center for Advanced Research/Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan. [3]Department of Biotechnology, Graduate School of Agricultural and Life Sciences, The University of Tokyo, Japan. [4]Bioinformatics Course, Master's/Doctoral Program in Life Science Innovation (T-LSI), School of Integrative and Global Majors (SIGMA), University of Tsukuba, Japan.

References

1. Trapnell, C.: Defining cell types and states with single-cell genomics. Genome Research **25(10)**, 1491–1498 (2015)
2. Macosko, E.Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A.R., Kamitaki, N., Martersteck, E.M., Trombetta, J.J., Weitz, D.A., Sanes, J.R., Shalek, A.K., Regev, A., McCarroll, S.A.: Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell **161**, 1202–1214 (2015)
3. Shekhar, K., Lapan, S.W., Whitney, I.E., Tran, N.M., Macosko, E.Z., Kowalczyk, M., Adiconis, Z., Levin, J.Z., Nemesh, J., Goldman, M., McCarroll, S.A., Cepko, C.L., Regev, A., Sanes, J.R.: Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. Cell **166**, 1308–1323 (2016)
4. Campbell, J.N., Macosko, E.Z., Fenselau, H., Pers, T.H., Lyubetskaya, A., Tenen, D., Goldman, M., Verstegen, A.M.J., Resch, J.M., McCarroll, S.A., Rosen, E.D., Lowell, B.B., Tsai, L.T.: A molecular census of arcuate hypothalamus and median eminence cell types. Nature Neuroscience **20(3)**, 484–496 (2017)
5. Klein, A.M., Mazutis, L., Akartuna, I., Tallapragada, N., Veres, A., Li, V., Peshkin, L., Weitz, D.A., Kirschner, M.W.: Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell **161**, 1187–1201 (2015)
6. Baron, M., Veres, A., Wolock, S.L., Faust, A.L., Gaujoux, R., Vetere, A., Ryu, J.H., Wagner, B.K., Shen-Orr, S.S., Klein, A.M., Melton, D.A., Yanai, I.: A single-cell transcriptomic map of the human and mouse pancreas reveals inter- and intra-cell population structure. Cell Systems **3(4)**, 346–360 (2016)
7. Grun, D., Lyubimova, A., Kester, L., Wiebrands, K., Basak, O., sasaki, N., Clevers, H., Oudenaarden, A.: Single-cell messenger rna sequencing reveals rare intestinal cell types. Nature **525**, 251–255 (2015)

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

8. Buettner, F., Natarajan, K.N., Casale, F.P., Proserpio, V., Scialdone, A., Theis, F.J., Teichmann, S.A., Marioni, J.C., Stegle, O.: Computational analysis of cell-to-cell heterogeneity in single-cell rna-sequencing data reveals hidden subpopulations of cells. Nature Biotechnology **33(2)**, 155–160 (2015)
9. Durruthy-Durruthy, R., Gottlieb, A., Hartman, B.H., Waldhaus, J., Laske, R.D., Altman, R., Heller, S.: Reconstruction of the mouse otocyst and early neuroblast lineage at single-cell resolution. Cell **157**, 1–15 (2014)
10. Achim, K., Pettit, J.B., Saraiva, L.R., Gavriouchkina, D., Larsson, T., Arendt, D., Marioni, J.C.: High-throughput spatial mapping of single-cell rna-seq data to tissue of origin. Nature Computational Biology **33(5)**, 503–509 (2015)
11. Satija, R., Farrell, J.A., Gennert, D., Schier, A.F., Regev, A.: Spatial reconstruction of single-cell gene expression data. Nature Biotechnology **33(5)**, 495–508 (2015)
12. Trapnell, C., Cacchiarelli, D., Grimsby, J., Pokhare, P., Li, S., Morse, M., Lennon, N.J., Livak, K.J., Mikkelsen, T.S., Rinn, J.L.: The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nature Biotechnology **32**, 381–386 (2014)
13. Qiu, X., Mao, Q., Tang, Y., Wang, L., Chawla, R., Pliner, H.A., Trapnell, C.: Reversed graph embedding resolves complex single-cell trajectories. Nature Methods **14(10)**, 979–982 (2017)
14. Svensson, V., Tormo, R.V., Teichmann, S.A.: Exponential scaling of single-cell rna-seq in the past decade. Nature Protocols **13(4)**, 599–604 (2017)
15. Sasagawa, Y., Danno, H., Takada, H., Ebisawa, M., Tanaka, K., Hayashi, T., Kurisaki, A., Nikaido, I.: Quartz-seq2: a high-throughput single-cell rna-sequencing method that effectively uses limited sequence reads. BMC Genome Biology **19(29)** (2018)
16. Jaitin, D.A., Kenigsberg, E., Keren-Shaul, H., Elefant, N., Paul, F., Zaretsky, I., Mildner, A., Cohen, N., Jung, S., Tanay, A., Amit, I.: Massively parallel single cell rna-seq for marker-free decomposition of tissues into cell types. Science **343(6172)**, 776–779 (2014)
17. Hashimshony, T., Senderovich, N., Avital, G., Klochendler, A., de Leeuw, Y., Anavy, L., Gennert, D., Li, S., Livak, K.L., Rozenblatt-Rosen, O., Dor, Y., Regev, A., Yanai, I.: Cel-seq2: sensitive highly-multiplexed single-cell rna-seq. BMC Genome Biology **17(77)** (2016)
18. Zeisel, A., Muñoz-Manchado, A.B., Codeluppi, S., Lönnerberg, P., Manno, G.L., Juréus, A., Marques, S., Munguba, H., He, L., Betsholtz, C., Rolny, C., Castelo-Branco, G., Hjerling-Leffler, J., Linnarsson, S.: Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. Science **347(6226)**, 1138–1142 (2015)
19. Hashimshony, T., Senderovich, N., Avital, G., Klochendler, A., de Leeuw, Y., Anavy, L., Gennert, D., Li, S., Livak, K.J., Rozenblatt-Rosen, O., Dor, Y., Regev, A., Yanai, I.: Cel-seq2: sensitive highly-multiplexed single-cell rna-seq. Genome Biology **17(77)** (2016)
20. Shalek, A.K., Satija, R., Shuga, J., Trombetta, J.J., Gennert, D., Lu, D., Chen, P., Gertner, R.S., Gaublomme, J.T., Yosef, N., Schwartz, S., Fowler, B., Weaver, S., Wang, J., Ding, R., Raychowdhury, R., Friedman, N., Hacohen, N., Park, H., May, A.P., Regev, A.: Single cell rna seq reveals dynamic paracrine control of cellular variation. Nature **510(7505)** (2014)
21. Tasic, B., Menon, V., Nguyen, T.N., Kim, T.K., Jarsky, T., Yao, Z., Levi, B., Gray, L.T., Sorensen, S.A., Dolbeare, T., Bertagnolli, D., Goldy, J., Shapovalova, N., Pary, S., Parry, C., Lee, C., Smith, K., Bernard, A., Madisen, L., Sunkin, S.M., Hawrylycz, M., Koch, C., Zeng, H.: Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nature Neuroscience **19(2)**, 335–346 (2016)
22. Zheng, G.X.Y., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., Gregory, M.T., Shuga, J., Montesclaros, L., Underwood, J.G., Masquelier, D.A., Nishimura, S.Y., Schnall-Levin, M., Wyatt, P.W., Hindson, C.M., Bharadwai, R., Wong, A., Ness, K.D., Beppu, L.W., Deeg, H.J., McFarland, C., Loeb, K.R., Valente, W.J., Ericson, N.G., Stevens, E.A., Radich, J.P., Mikkelsen, T.S., Hindson, B.J., Bielas, J.H.: Massively parallel digital transcriptional profiling of single cells. Nature Communications **8(14049)**, 1–12 (2017)
23. Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D.M., Hill, A.J., Zhang, F., Mundlos, S., Christiansen, L., Steemers, F.J., Trapnell, C., Shendure, J.: The single-cell transcriptional landscape of mammalian organogenesis. Nature (2019)
24. Consortium, T.H.: The human cell atlas white paper (2017)
25. Rozenblatt-Rosen, O., Stubbington, M.J.T., Regev, A., Teichmann, S.A.: The human cell atlas: from vision to reality. Nature **550**, 451–453 (2017)
26. Regev, A., Teichmann, S.A., Lander, E.S., Amit, I., Benoist, C., Birney, E., Bodenmiller, B., Campbell, P., Carninci, P., Clatworthy, M., Clevers, H., Deplancke, B., Dunham, I., Eberwine, J., Eils, R., Enard, W., Farmer, A., Fugger, L., Göttgens, B., Hacohen, N., Haniffa, M., Hemberg, M., Kim, S., Klenerman, P., Kriegstein, A., Lein, E., Linnarsson, S., Lundberg, E., Lundeberg, J., Majumder, P., Marioni, J.C., Merad, M., Mhlanga, M., Nawijn, M., Netea, M., Nolan, G., Pe'er, D., Phillipakis, A., Ponting, C.P., Quake, S., Reik, W., Rozenblatt-Rosen, O., Sanes, J., Satija, R., Schumacher, T.N., Shalek, A., Shapiro, E., Sharma, P., Shin, J.W., Stegle, O., Stratton, M., Stubbington, M.J.T., Theis, F.J., Uhlen, M., van Oudenaarden, A., Wagner, A., Watt, F., Weissman, J., Wold, B., Xavier, R., Yosef, N., Participants, H.C.A.M.: Science forum: The human cell atlas. eLife, 37041 (2017)
27. Han, X., Wang, R., Zhou, Y., Fei, L., Sun, H., Lai, S., Saadatpour, A., Zhou, Z., Chen, H., Ye, F., Huang, D., Xu, Y., Huang, W., Jiang, M., Jiang, X., Mao, J., Chen, Y., Lu, C., Xie, J., Fang, Q., Wang, Y., Yue, R., Li, T., Huang, H., Orkin, S.H., Yuan, G.C., Chen, M., Guo, G.: Mapping the mouse cell atlas by microwell-seq. Cell **172(5)**, 1091–1107 (2018)
28. Consortium, T.T.M.: Single-cell transcriptomics of 20 mouse organs creates a tabula muris. Nature **562(7727)**, 367–372 (2018)
29. Wagner, A., Regev, A., Yosef, N.: Revealing the vectors of cellular identity with single-cell genomics. Nature Biotechnology **34(11)**, 1145–160 (2017)
30. Stegle, O., Teichmann, S.A., Marioni, J.C.: Computational and analytical challenges in single-cell

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

transcriptomics. Nature Reviews Genetics **16(3)**, 133–145 (2015)
31. Bacher, R., Kendziorski, C.: Design and computational analysis of single-cell rna-sequencing experiments. BMC Genome Biology **17(63)** (2016)
32. Poulin, J.F., Tasic, B., Hjerling-Leffler, J., Trimarchi, J.M., Awatramani, R.: Disentangling neural cell diversity using single-cell transcriptomics. Nature Neuroscience **19(9)**, 1131–1141 (2016)
33. Kolodziejczyk, A.A., Kim, J.K., Svensson, V., Marioni, J.C., Teichmann, S.A.: The technology and biology of single-cell rna sequencing. Molecular Cell **58(4)**, 610–620 (2015)
34. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine **2(11)**, 559–572 (1901)
35. Hotelling, H.: Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology **24**, 417–441498520 (1933)
36. Broa, R., K, S.A.: Principal component analysis. Royal Society of Chemistry **6(2812)**, 2812–2831 (2014)
37. Tasic, B., Menon, V., Nguyen, T.N., Kim, T.K., Jarsky, T., Yao, Z., Levi, B., Gray, L.T., Sorensen, S.A., Dolbeare, T., Bertagnolli, D., Goldy, J., Shapovalova, N., Parry, S., Lee, C., Smith, K., Bernard, A., Madisen, L., Sunkin, S.M., Hawrylycz, M., Koch, C., Zeng, H.: Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nature Neuroscience **19(2)**, 335–346 (2016)
38. Baron, M., Veres, A., Wolock, S.L., Faust, A.L., Gaujoux, R., Vetere, A., Ryu, J.H., Wagner, B.K., Shen-Orr, S.S., Klein, A.M., Melton, D.A., Yanai, I.: A single-cell transcriptomic map of the human and mouse pancreas reveals inter- and intra-cell population structure. Cell Systems **3(4)**, 346–360 (2016)
39. Genomics, X.: 1.3 Million Brain Cells from E18 Mice. https://support.10xgenomics.com/single-cell/datasets/1M_neurons
40. Cole, M.B., Risso, D., Wagner, A., DeTomaso, D., Ngai, J., Purdom, E., Dudoit, S., Yosef, N.: Performance assessment and selection of normalization procedures for single-cell rna-seq. Cell Systems **8(4)**, 315–328 (2019)
41. Taguchi, Y.-H.: Principal component analysis-based unsupervised feature extraction applied to single-cell gene expression analysis. In: 14th International Conference, ICIC 2018, pp. 816–826 (2018). China
42. Lin, Z., Yang, C., Zhu, Y., Duchi, J., Fu, Y., Wang, Y., Jiang, B., Zamanighomi, M., Xu, X., Li, M., Sestan, N., Zhao, H., Wong, W.H.: Simultaneous dimension reduction and adjustment for confounding variation. PNAS **113(51)**, 14662–14667 (2016)
43. Lasrado, R., Boesmans, W., Kleinjung, J., Pin, C., Bell, D., Bhaw, L., McCallum, S., Zong, H., Luo, L., Clevers, H., Vanden, B.P., Pachnis, V.: Lineage-dependent spatial and functional organization of the mammalian enteric nervous system. Science **356(6339)**, 722–726 (2017)
44. Wagner, F.: Go-pca: An unsupervised method to explore gene expression data using prior knowledge. PLOS ONE **10(11)**, 0143196 (2015)
45. Cerosaletti, K., Barahmand-Pour-Whitman, F., Yang, J., DeBerg, H.A., Dufort, M.J., Murray, S.A., Israelsson, E., Speake, C., Gersuk, V.H., Eddy, J.A., Reijonen, H., Greenbaum, C.J., Kwok, W.W., Wambre, E., Prlic, M., Gottardo, R., Nepom, G.T., Linsley, P.S.: Single-cell rna sequencing reveals expanded clones of islet antigen-reactive cd4+ t cells in peripheral blood of subjects with type 1 diabetes. Journal of Immunology **199(1)**, 323–325 (2017)
46. Single-cell transcriptomes reveal characteristic features of human pancreatic islet cell types. EMBO Reports **17(2)**, 178–187 (2016)
47. Butler, H.P. A, Smibert, P., Papalexi, E., Satija, R.: Integrated analysis of single cell transcriptomic data across conditions, technologies, and species. Nature Biotechnology **36**, 411–420 (2018)
48. Lun, A.T., McCarthy, D.J., Marioni, J.C.: A step-by-step workflow for low-level analysis of single-cell rna-seq data with bioconductor. F1000Research **Version2** (2016)
49. Ilicic, T., Kim, J.K., Kolodziejczyk, A.A., Bagger, F.O., McCarthy, D.J., Marioni, J.C., Teichmann, S.A.: Classification of low quality cells from single-cell rna-seq data. BMC Genome Biology **17(29)** (2016)
50. Dijk, D., Sharma, R., Nainys, J., Yim, K., Kathail, P., Carr, A.J., Burdziak, C., Moon, K.R., Chaffer, C.L., Pattabiraman, D., Bierie, B., Mazutis, L., Wolf, G., Krishnaswamy, S., Pe'er, D.: Recovering gene interactions from single-cell data using data diffusion. Cell **174(3)**, 716–729 (2018)
51. Li, W.V., Li, J.J.: An accurate and robust imputation method scimpute for single-cell rna-seq data. Nature Communication **9(997)** (2018)
52. Gong, W., Kwak, I.Y., Pota, P., Koyano-Nakagawa, N., Garry, D.J.: Drimpute: imputing dropout events in single cell rna sequencing data. BMC Bioinformatics **19(220)** (2018)
53. Büttner, M., Miao, Z., Wolf, F.A., Teichmann, S.A., Theis, F.J.: A test metric for assessing single-cell rna-seq batch correction. Nature methods **16(1)**, 43–49 (2019)
54. Shaham, U., Stanton, K.P., Zhao, J., Li, H., Raddassi, K., Montgomery, R., Kluger, Y.: Removal of batch effects using distribution-matching residual networks. Bioinformatics **33(16)**, 2539–2546 (2017)
55. Korsunsky, I., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.-R., Raychaudhuri, S.: Fast, sensitive, and accurate integration of single cell data with harmony. bioRxiv (2018). doi:10.1101/461954
56. Scialdone, A., Natarajan, K.N., Saraiva, L.R., Proserpio, V., Teichmann, S.A., Stegle, O., Marioni, J.C., Buettner, F.: Computational assignment of cell-cycle stage from single-cell transcriptome data. Methods **85**, 54–61 (2015)
57. Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Reseach, 2579–2605 (2008)
58. Maaten, L.: Accelerating t-sne using tree-based algorithms. Journal of Machine Learning Reseach, 3221–3245 (2014)
59. Linderman, G.C., Rachh, M., Hoskins, J.G., Steinerberger, S., Kluger, Y.: Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. Nature methods **16**, 243–245 (2019)
60. Lawrence, N.D.: Gaussian process latent variable models for visualisation of high dimensional data. In: In NIPS, p. 2004 (2003)
61. McInnes, L., Healy, J., Saul, N., Großberger, L.: Umap: Uniform manifold approximation and projection for

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

dimension reduction. The Journal of Open Source Software **3(29)**, 861 (2018)
62. Becht, E., McInnes, L., Healy, J., Dutertre, C.A., Kwok, I.W.H., Ng, L.G., Ginhoux, F., Newell, E.W.: Dimensionality reduction for visualizing single-cell data using umap. Nature Biotechnology **37**, 38–44 (2019)
63. Weinreb, C., Wolock, S., Klein, A.M.: Spring: a kinetic interface for visualizing high dimensional single-cell expression data. Bioinformatics **34(7)**, 1246–1248 (2018)
64. Kiselev, V.Y., Kirschner, K., Schaub, M.T., Andrews, T., Yiu, A., Chandra, T., Natarajan, K.N., Reik, W., Barahona, M., Green, A.R., Hemberg, M.: Sc3: consensus clustering of single-cell rna-seq data. Nature methods **14(5)**, 483–486 (2017)
65. Wang, B., Zhu, J., Pierson, E., Ramazzotti, D., Batzoglou, S.: Visualization and analysis of single-cell rna-seq data by kernel-based similarity learning. Nature methods **14(4)**, 414–416 (2017)
66. Yang, Y., Huh, R., Culpepper, H.W., Lin, Y., Love, M.I., Li, Y.: Safe-clustering: Single-cell aggregated (from ensemble) clustering for single-cell rna-seq data. Bioinformatics (2018)
67. Zurauskiene, J., Yau, C.: pcareduce: hierarchical clustering of single cell transcriptional profiles. BMC Bioinformatics **17(140)** (2016)
68. Tsoucas, D., Yuan, G.C.: Giniclust2: a cluster-aware, weighted ensemble clustering method for cell-type detection. BMC Genome Biology **19(1)** (2018)
69. Herman, J.S., Sagar, Grün, D.: Fateid infers cell fate bias in multipotent progenitors from single-cell rna-seq data. Nature methods **15**, 379–386 (2018)
70. Sato, K., Tsuyuzaki, K., Shimizu, K., Nikaido, I.: Cellfishing.jl: an ultrafast and scalable cell search method for single-cell rna sequencing. BMC Genome Biology **20(1)** (2019)
71. Diaz, A., Liu, S.J., Sandoval, C., Pollen, A., Nowakowski, T.J., Lim, D.A., Kriegstein, A.: Scell: integrated analysis of single-cell rna-seq data. Bioinformatics **32(14)**, 2219–2220 (2016)
72. Ji, Z., Ji, H.: Tscan: Pseudo-time reconstruction and evaluation in single-cell rna-seq analysis. Nucleic Acids Research **44(13)** (2016)
73. Shin, J., Berg, D.A., Zhu, Y., Shin, J.Y., Song, J., Bonaguidi, M.A., Enikolopov, G., Nauen, D.W., Christian, K.M., Ming, G.L., Song, H.: Single-cell rna-seq with waterfall reveals molecular cascades underlying adult neurogenesis. Cell Stem Cell **17(3)**, 360–372 (2015)
74. Street, K., Risso, D., Fletcher, R.B., Das, D., Ngai, J., Yosef, N., Purdom, E., Dudoit, S.: Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. BMC Genomics **19(477)** (2018)
75. Campbell, K.R., Yau, C.: Probabilistic modeling of bifurcations in single-cell gene expression data using a bayesian mixture of factor analyzers. Wellcome Open Research **2(19)** (2017)
76. McCarthy, D.J., Campbell, K.R., Lun, A.T., Wills, Q.F.: Scater: pre-processing, quality control, normalization and visualization of single-cell rna-seq data in r. Bioinformatics **33(8)**, 1179–1186 (2017)
77. Jenkins, D., Faits, T., Khan, M.M., Briars, E., Carrasco, P.S., Johnson, W.E.: singleCellTK: Interactive Analysis of Single Cell RNA-Seq Data. https://bioconductor.org/packages/release/bioc/html/singleCellTK.html (2018)
78. Tian, L., Su, S., Dong, X., Amann-Zalcenstein, D., Biben, C., Seidi, A., Hilton, D.J., Naik, S.H., Ritchie, M.E.: scpipe: A flexible r/bioconductor preprocessing pipeline for single-cell rna-sequencing data. PLOS Computational Biology **14(8)**, 1006361 (2018)
79. Yip, S.H., Wang, P., Kocher, J.A., Sham, P.C., Wang, J.: Linnorm: improved statistical analysis for single cell rna-seq expression data. Nucleic Acids Research **45(22)**, 179 (2017)
80. Finak, G., McDavid, A., Yajima, M., Deng, J., Gersuk, V., Shalek, A.K., Slichter, C.K., Miller, H.W., McElrath, M.J., Prlic, M., Linsley, P.S., Gottardo, R.: Mast: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell rna sequencing data. BMC Genome Biology **16(278)** (2015)
81. Demsar, J., Curk, T., Erjavec, A., Gorup, C., Hocevar, T., Milutinovic, M., Mozina, M., Polajnar, M., Toplak, M., Staric, A., Stajdohar, M., Umek, L., Zagar, L., Zbontar, J., Zitnik, M., Zupan, B.: Orange: Data mining toolbox in python. Journal of Machine Learning Research, 2349–2353 (2013)
82. Zhu, X., Wolfgruber, T.K., Tasato, A., Arisdakessian, C., Garmire, D.G., Garmire, L.X.: Granatum: a graphical single-cell rna-seq analysis pipeline for genomics scientists. BMC Genome Medicine **9(108)** (2017)
83. Azizi, E., Carr, A.J., Plitas, G., Cornish, A.E., Konopacki, C., Prabhakaran, S., Nainys, J., Wu, K., Kiseliovas, V., Setty, M., Choi, K., Fromme, R.M., Dao, P., McKenney, P.T., Wasti, R.C., Kadaveru, K., Mazutis, L., Rudensky, A.Y., Pe'er, D.: Single-cell map of diverse immune phenotypes in the breast tumor microenvironment. Cell **5(23)**, 1293–1308 (2018)
84. Golub, G.H., Loan, C.F.V.: Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences), Fourth Edition. Johns Hopkins University Press, ??? (2012)
85. Senabouth, A., Lukowski, S., Alquicira, J., Andersen, S., Mei, X., Nguyen, Q., Powell, J.: ascend: R package for analysis of single cell rna-seq data. bioRxiv (2017). doi:10.1101/207704
86. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondl, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. The Journal of Machine Learning Research **12**, 2825–2830 (2011)
87. Wolf, F.A., Angerer, P., Theis, F.J.: Scanpy: large-scale single-cell gene expression data analysis. BMC Genome Biology **19(15)** (2018)
88. Bhaduri, A., Nowakowski, T.J., Pollen, A.A., Kriegstein, A.R.: Identification of cell types in a mouse brain single-cell atlas using low sampling coverage. BMC Biology (2018)
89. Levy, A., M, K.: Sequential karhunen-loeve basis extraction and its application to images. IEEE Transactions on Image Processing **9(8)**, 1371–1374 (2000)
90. Bai, Z., Demmel, J., Dongarra, J., Ruhe, A., Vorst, H.V.D.: Templates for the Solution of Algebraic Eigenvalue Problems, A Practical Guide. Society for Industrial and Applied Mathematics, ??? (1987)
91. Lehoucq, R., Maschhoff, K., Sorensen, D., Yang, C.: ARPACK SOFTWARE. https://www.caam.rice.edu/software/ARPACK/
92. Qiu, Y.: Spectra: C++ Library For Large Scale Eigenvalue Problems. https://spectralib.org

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

93. Larsen, R.M.: PROPACK homepage. http://sun.stanford.edu/~rmunk/PROPACK/
94. Baglama, J., Reichel, L.: Augmented implicitly restarted lanczos bidiagonalization methods. SIAM Journal on Scientific Computing **27(1)**, 19–42 (2005)
95. Lehoucq, R.B., Sorensen, D.C., Yang, C.: Arpack users' guide: Solution of large-scale eigenvalue problems with implicitly restarted arnoldi methods (1997)
96. Chen, J., Noack, A., Edelman, A.: Fast computation of the principal components of genotype matrices in julia. arXiv (2018). doi:arXiv:1808.03374v1
97. Balzano, L., Chi, Y., Lu, Y.M.: Streaming pca and subspace tracking: The missing data case. Proceedings of the IEEE **106(8)**, 1293–1310 (2018)
98. Oja, E.: A simplified neuron model as a principal component analyzer. Journal of Mathematical Biology **15**, 267–273 (1982)
99. Oja, E., Karhunen, J.: On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix author links open overlay panel. Journal of Mathematical Analysis and Applications **106(1)**, 69–84 (1985)
100. Oja, E.: Principal components, minor components, and linear neural networks. Neural Networks **5**, 927–935 (1992)
101. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev., Survey and Review **53(2)**, 217–288 (2011)
102. Halko, N., Martinsson, P.G., Shkolnisky, Y., M, T.: An algorithm for the principal component analysis of large data sets. SIAM Journal on Scientific Computing **33(5)**, 2580–2594 (2011)
103. Li, H., C, L.G., Szlam, A., Stanton, K.P., Kluger, Y., Tygert, M.: Algorithm 971: An implementation of a randomized algorithm for principal component analysis. ACM Transactions on Mathematical Software **43(3)** (2017)
104. Abraham, G., Inouye, M.: Fast principal component analysis of large-scale genome-wide data. PLOS ONE **9(4)**, 93766 (2014)
105. Lacono, G., Mereu, E., Guillaumet-Adkins, A., Corominas, R., Cusco, I., Rodriguez-Esteban, G., Gut, M., Perez-Jurado, L.A., Gut, I., Heyn, H.: bigscale: an analytical framework for big-scale single-cell data. Genome Research **28(6)**, 878–890 (2018)
106. Aibar, S., Gonzalez-Blas, C.B., Moerman, T., Huynh-Thu, V.A., Imrichova, H., Hulselmans, G., Rambow, F., Marine, J.-C., Geurts, P., Aerts, J., Oord, J., Atak, Z.K., Wouters, J., Aerts, S.: Scenic: single-cell regulatory network inference and clustering. Nature methods **14**, 1083–1086 (2017)
107. Crow, M., Paul, A., Ballouz, S., Huang, Z.J., Gilllis, J.: Characterizing the replicability of cell types defined by single cell rna-sequencing data using metaneighbor. Nature communications **884** (2018)
108. Kisekev, V.Y., Yiu, A., Hemberg, M.: scmap: projection of single-cell rna-seq data across data sets. Nature methods **15**, 359–362 (2018)
109. Huang, M., Wang, J., Torre, E., Dueck, H., Shaffer, S., Bonasio, R., Murray, J.I., Raj, A., Li, M., Zhang, N.R.: Saver: gene expression recovery for single-cell rna sequencing. Nature methods **15**, 539–542 (2018)
110. Wang, D., Gu, J.: Vasc: Dimension reduction and visualization of single-cell rna-seq data by deep variational autoencoder. Genomics, Proteomics & Bioinformatics **16(5)**, 320–331 (2018)
111. Ding, J., Condon, A., Shah, S.P.: Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. Nature Communications **2002** (2018)
112. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, ??? (2006)
113. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2(1)**, 193–218 (1985)
114. Rocklin, M.: Dask: Parallel computation with blocked algorithms and task scheduling. In: Huff, K., Bergstra, J. (eds.) Proceedings of the 14th Python in Science Conference, pp. 130–136 (2015)
115. Benson, A.R., Gleich, D.F., Demmel, J.: Direct qr factorizations for tall-and-skinny matrices in mapreduce architectures. Proceedings of the IEEE International Conference on Big Data (2013). doi:10.1109/BigData.2013.6691583
116. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42(8)**, 30–37 (2009)
117. Davis, T.: University of Florida Sparse Matrix Collection. https://sparse.tamu.edu
118. Yip, S.H., Sham, P.C., J, W.: Evaluation of tools for highly variable gene discovery from single-cell rna-seq data. Briefing in Bioinformatics, 011 (2018)
119. Erichson, N.B., Voronin, S., Brunton, S.L., Kutz, J.N.: Randomized matrix decompositions using r. arXiv (2016). doi:arXiv:1608.02148v4
120. Mineiro, P., Karampatziakis, N.: A randomized algorithm for cca. arXiv (2014). doi:arXiv:1411.3409v1
121. Arora, R., Cotter, A., Livescu, K., Srebro, N.: Stochastic optimization for pca and pls. In: 2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 861–868 (2012)

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

Figures

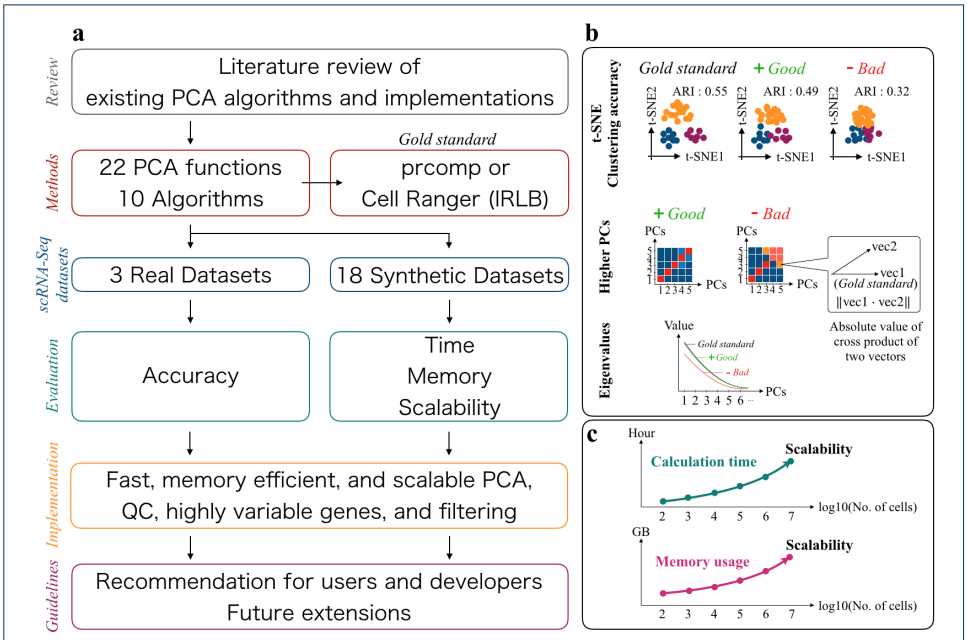

Figure 1 Overview of benchmarking in this work. (a) Schematic overview of this work. (b) Evaluation metrics of the benchmarking with real-world datasets. (c) Evaluation metrics of the benchmarking with synthetic datasets.

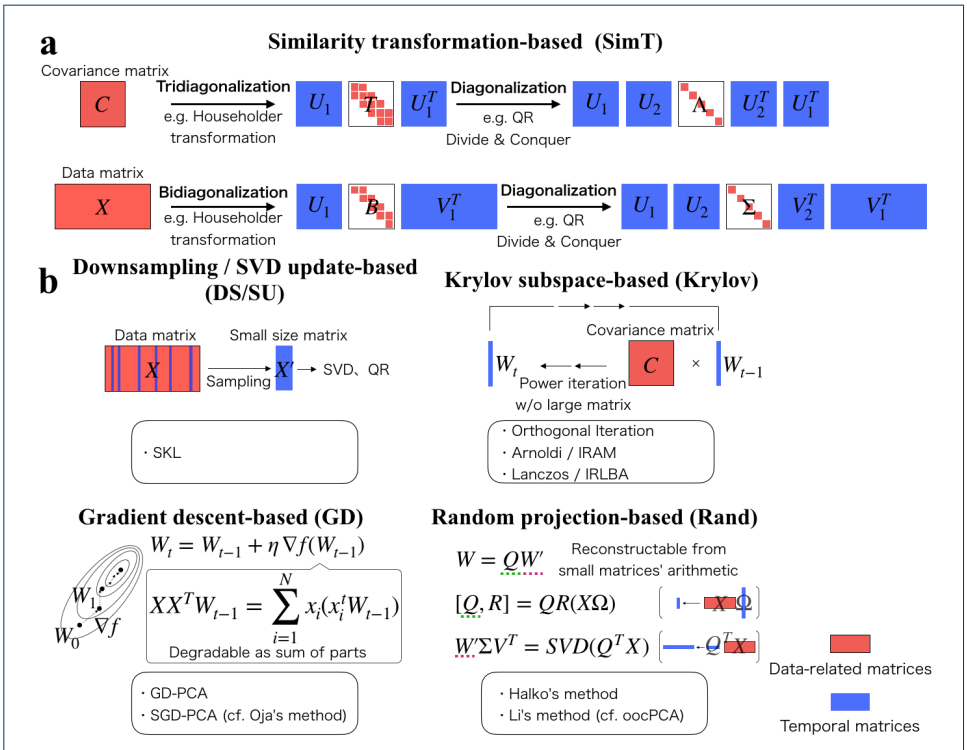

Figure 2 Some fast and memory-efficient PCA algorithms. (a) Full-rank SVD based on LAPACK and (b) truncated SVD based on the recent fast and memory-efficient PCA algorithms.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

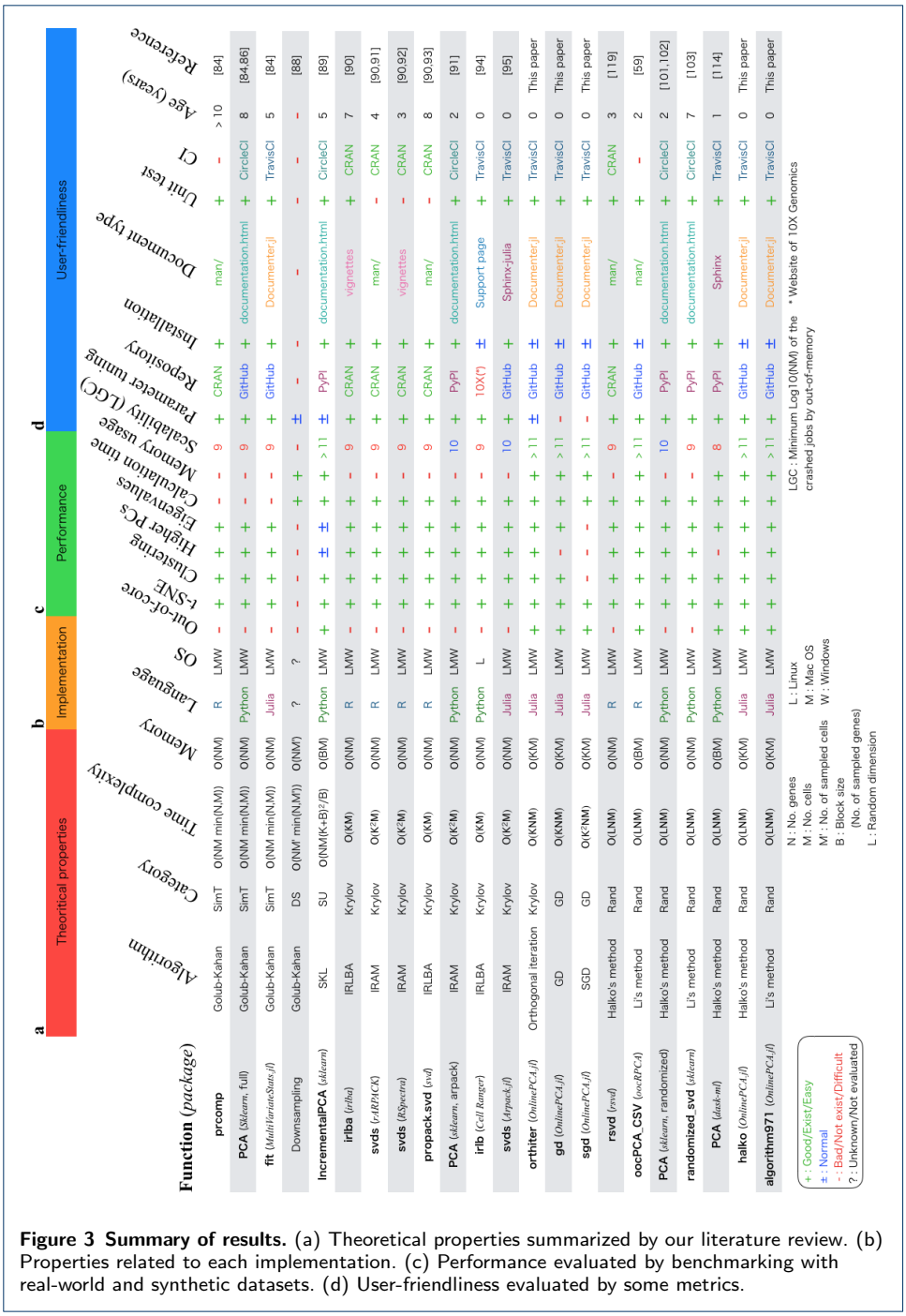

Figure 3 Summary of results. (a) Theoretical properties summarized by our literature review. (b) Properties related to each implementation. (c) Performance evaluated by benchmarking with real-world and synthetic datasets. (d) User-friendliness evaluated by some metrics.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

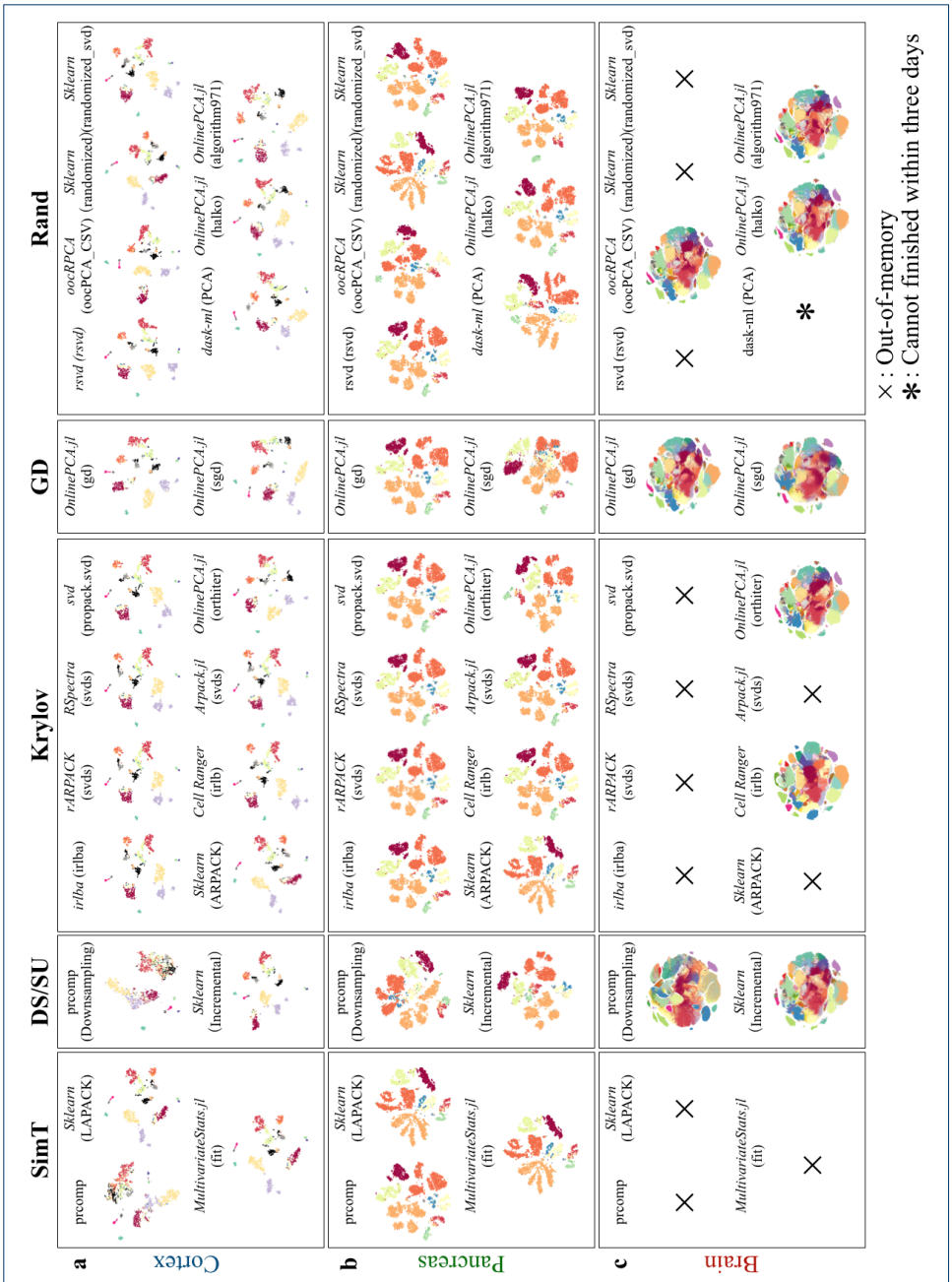

Figure 4 The comparison of t-SNE plots. Comparison of multiple PCA implementations performed with empirical datasets: (a) Cortex (10^3 cells), (b) Pancreas (10^4 cells), and (c) Brain datasets (10^6 cells). t-SNE was performed with the result of each PCA implementation. The cluster labels are the same as those of the respective original papers.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

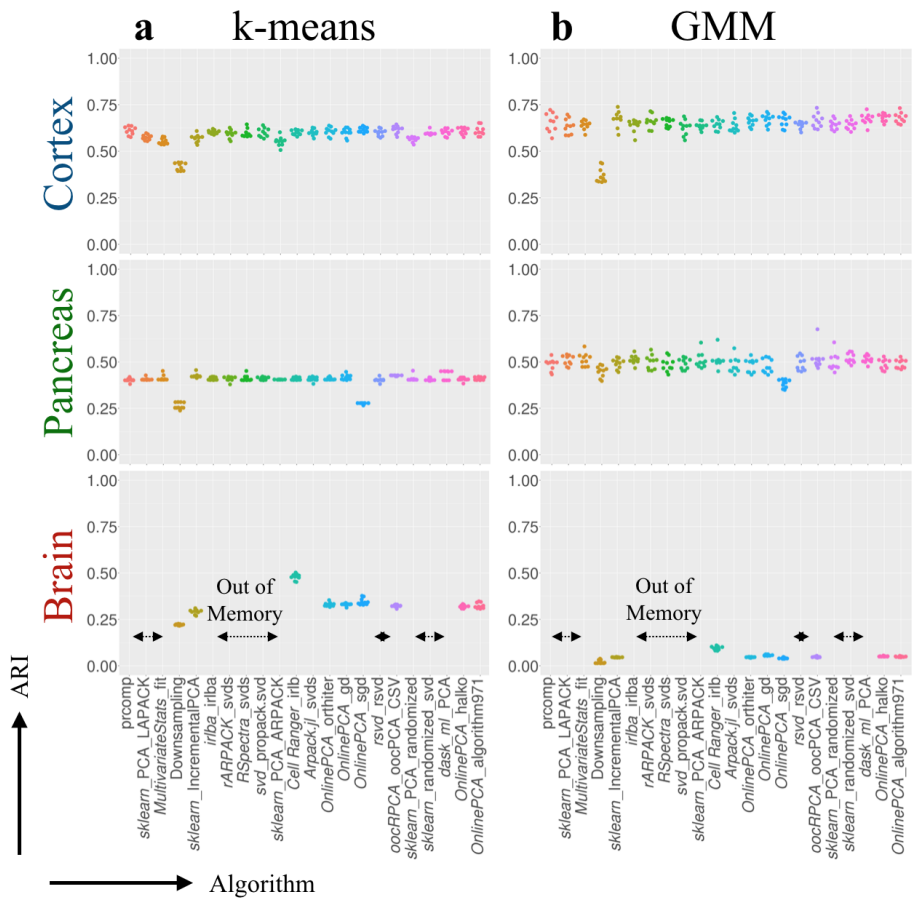

Figure 5 Clustering accuracy comparison. Clustering accuracy was evaluated by ARI of the result of k-means and GMM clustering. Multiple PCA implementations were performed for Cortex (10^3 cells), Pancreas (10^4 cells), and Brain datasets (10^6 cells), and k-means and GMM were performed for the PCA result. For each PCA result, k-means and GMM calculations were performed ten times. The cluster labels are the same as those of the respective original papers.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

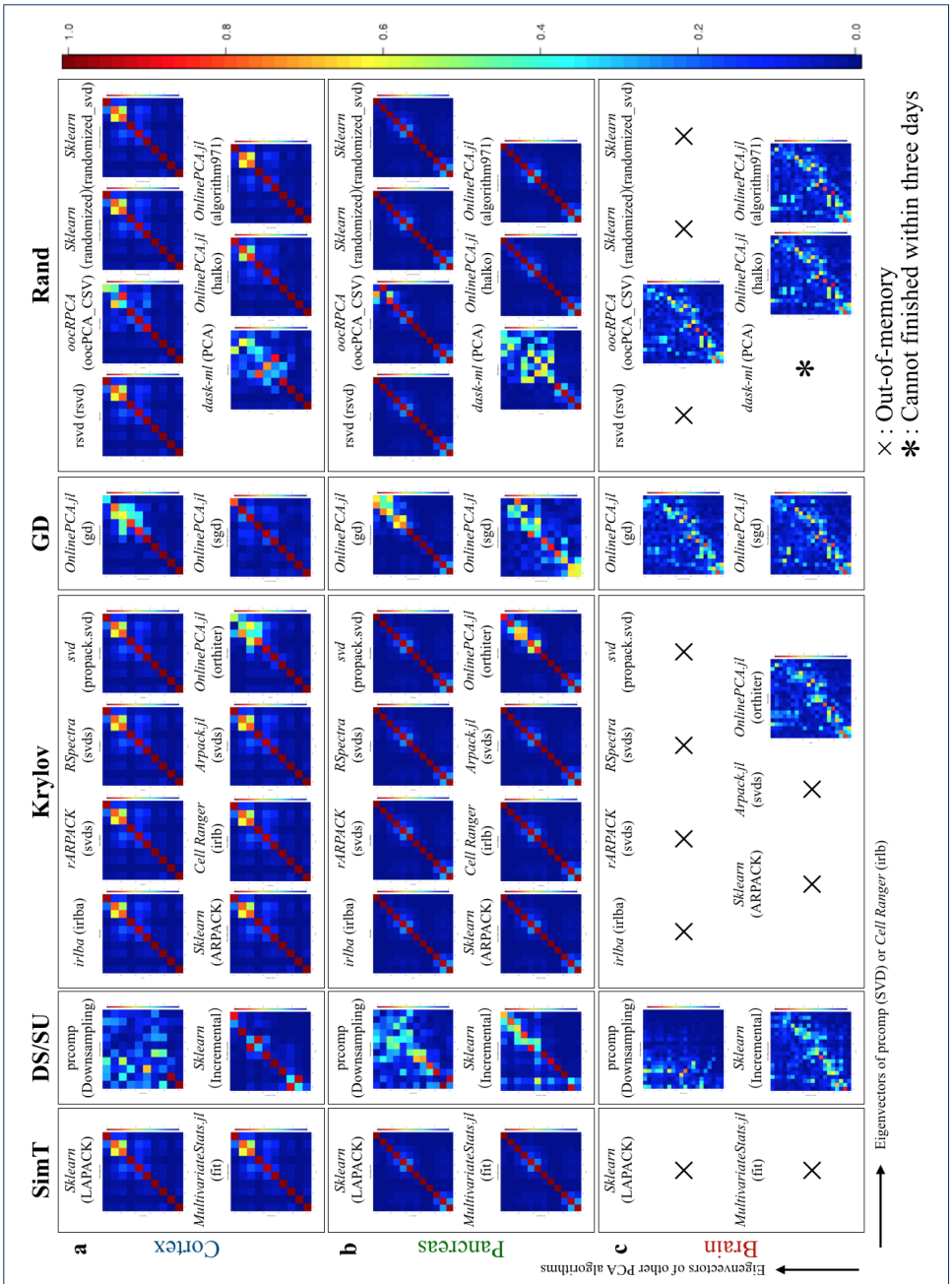

Figure 6 Comparison of all combinations of eigenvectors. Absolute values of the cross product of all combinations between the eigenvectors of the gold standard methods and those of the other PCA implementations were calculated. The closer the value is to 1, the closer the two corresponding eigenvectors are to each other. If two PCA results are equal without considering differences in sign, the matrix in this figure becomes an identical matrix.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

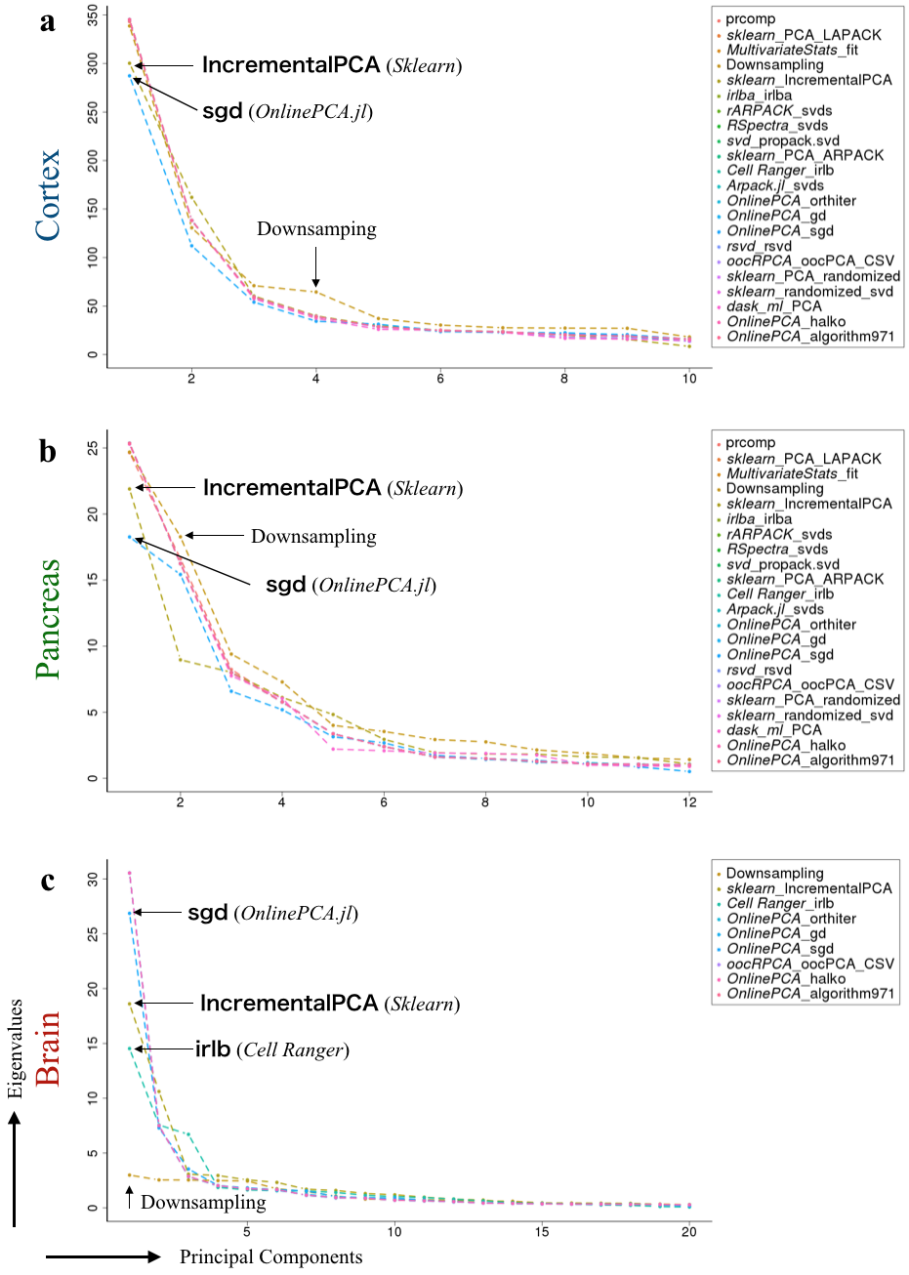

Figure 7 Comparison of eigenvalues. Distribution of eigenvalues of all the PCA implementations are drawn in each real dataset.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

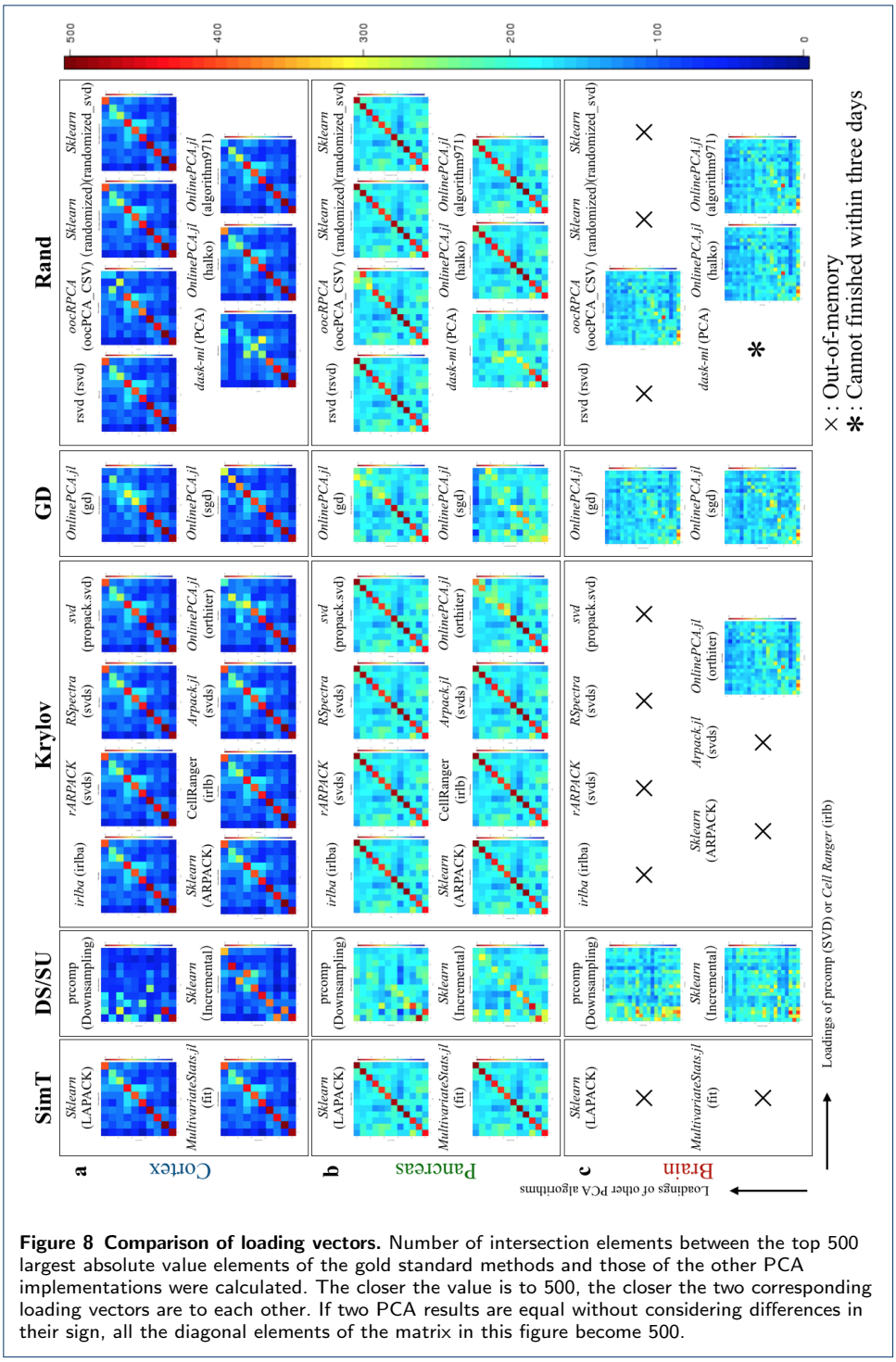

Figure 8 Comparison of loading vectors. Number of intersection elements between the top 500 largest absolute value elements of the gold standard methods and those of the other PCA implementations were calculated. The closer the value is to 500, the closer the two corresponding loading vectors are to each other. If two PCA results are equal without considering differences in their sign, all the diagonal elements of the matrix in this figure become 500.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

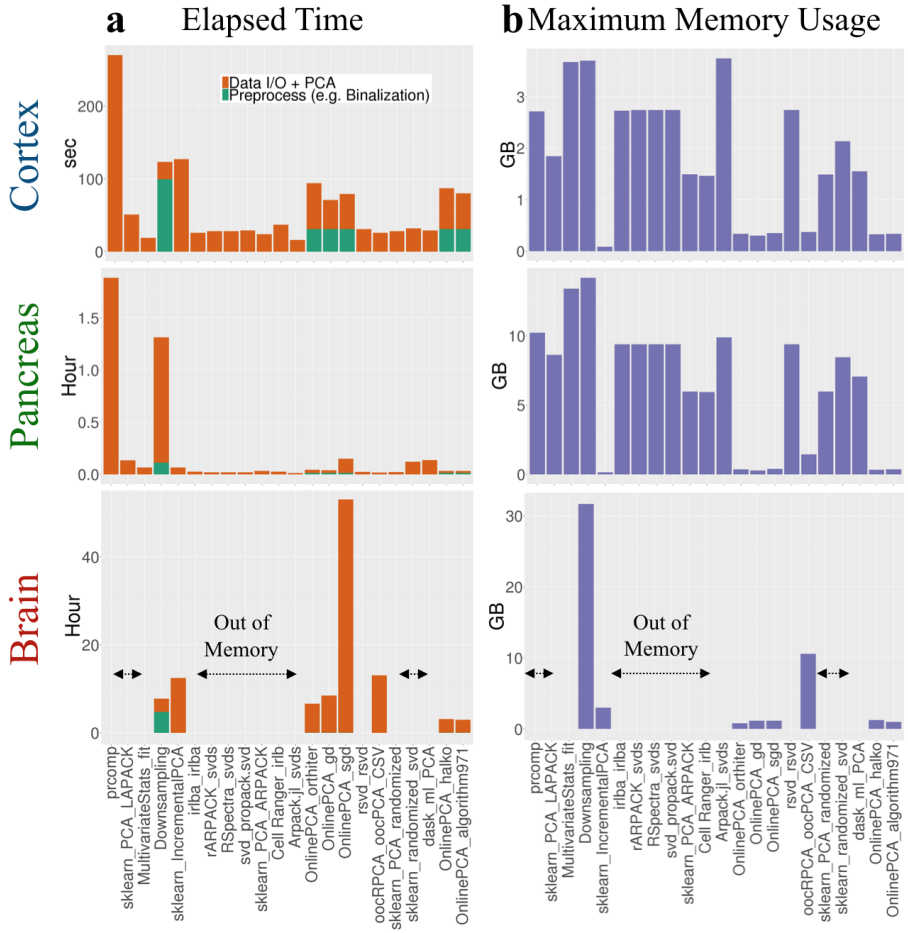

Figure 9 Comparison of the elapsed time and maximum memory usage for empirical datasets. (a) Elapsed time and (b) memory usage of all PCA implementations calculated for each empirical dataset.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

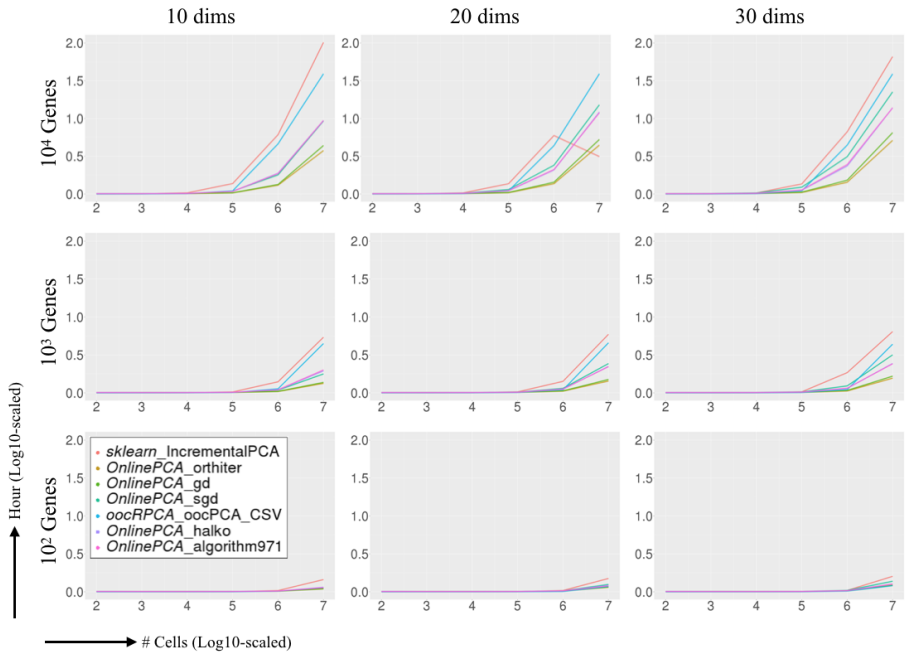

Figure 10 Comparison of the elapsed time for simulated datasets. Synthetic datasets ($\{10^2,10^3,10^4\}$ genes $\times$ $\{10^2,10^3,10^4,10^4,10^5,10^6,10^7\}$ cells matrices) were randomly generated, and all the out-of-core PCA implementations were performed. In each panel, the logarithm of the number of cells is indicated along the x-axis, and the elapsed time (hours) is shown along the y-axis.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

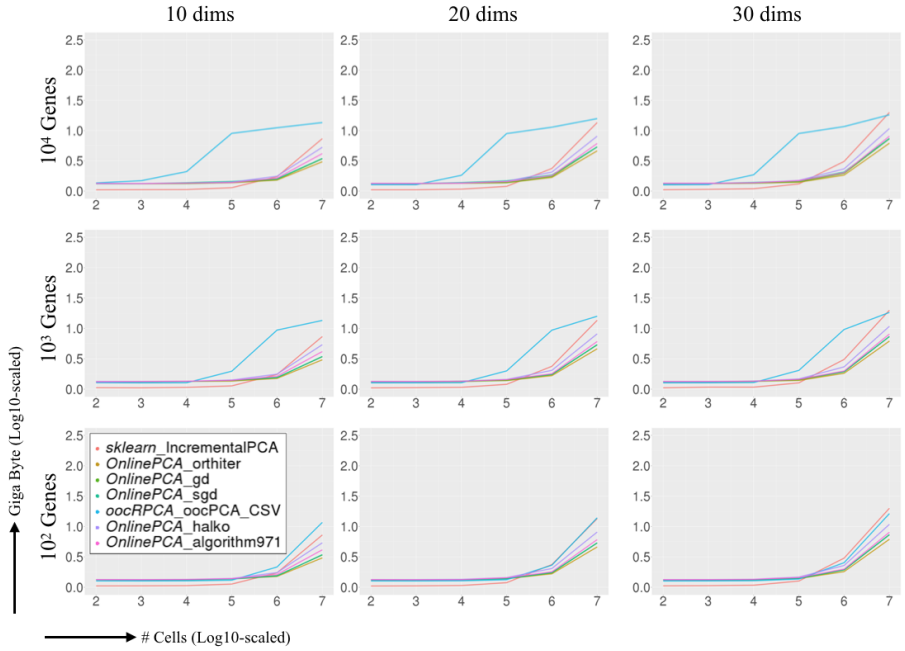

Figure 11 Comparison of the maximum memory usage for simulated datasets. Synthetic datasets ($\{10^2,10^3,10^4\}$ genes $\times$ $\{10^2,10^3,10^4,10^4,10^5,10^6,10^7\}$ cells matrices) were randomly generated, and all the out-of-core PCA implementations were performed. In each panel, the logarithm of the number of cells is indicate along the x-axis, and memory usage (GB) is shown along y-axis.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

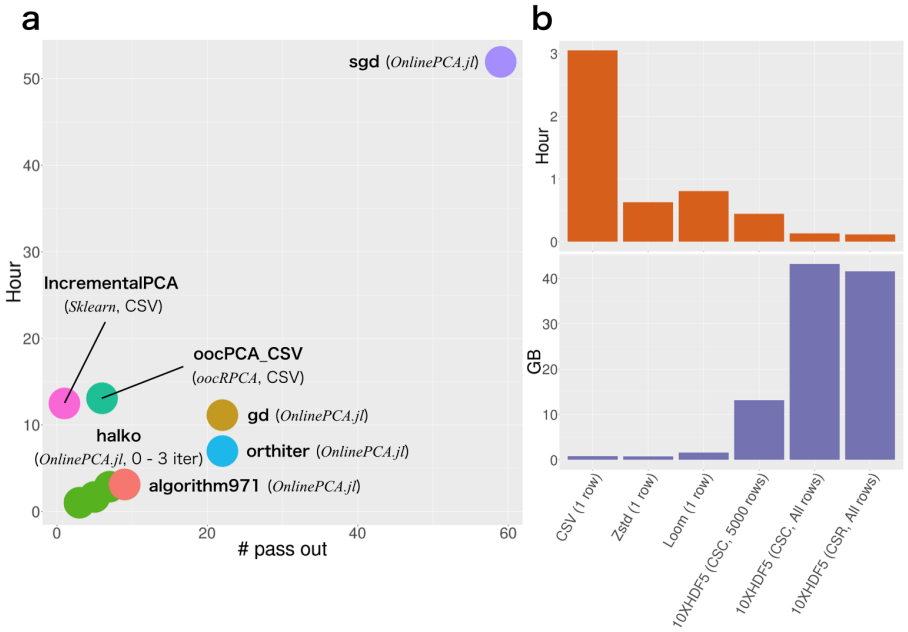

Figure 12 Relationship of the algorithms/implementations, the number of pass out, and the file format with the elapsed time for performing PCA with the Brain dataset. (a) Number of pass out for the data matrix and the calculation time for each algorithms/implementations were calculated. (b) Elapsed time and memory usage of one-pass orthogonal iteration were calculated.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

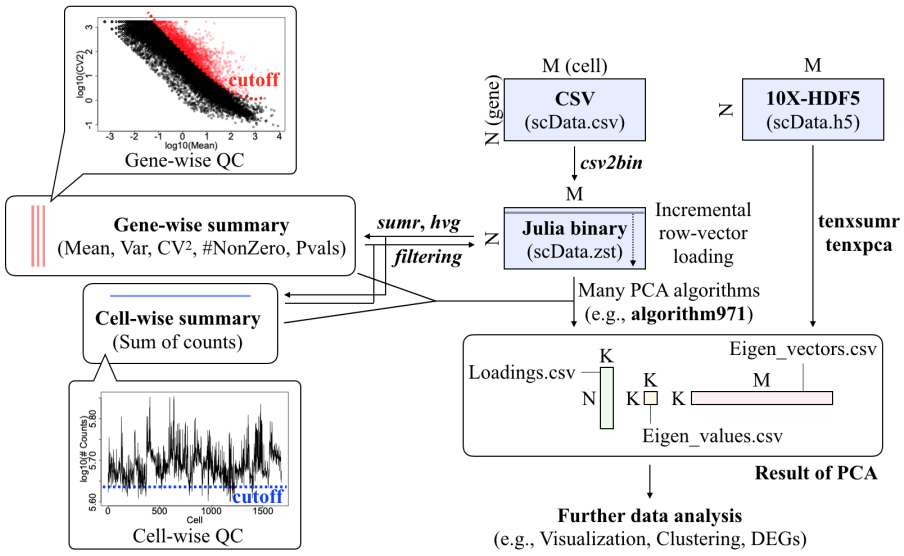

Figure 13 OnlinePCA.jl schematic. Input CSV files were first saved as a binary file with the `csv2bin` command and used with the `gd` and `sgd` command, which performs incremental PCA. When using the HDF5 file defined by 10X Genomics, we converted the file to CSV format using an in-house Python script. Gene-wise or cell-wise summary statistics were calculated using the `sumr` command. Highly variable genes can also be calculated by the `hvg` command. Because the gene-wise and cell-wise summary statistics are expressed as small vectors, it can be used to perform precise data quality control (QC) with any programming language. After QC, the `filtering` command removed low-quality genes and cells using a user-specified index. Combined with the small size vectors, some out-of-core PCA implementations, such as `orthiter`/`gd`/`sgd`/`halko`/`algorithm971` have commands to incrementally update eigenvectors from the row vector of the data matrix.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

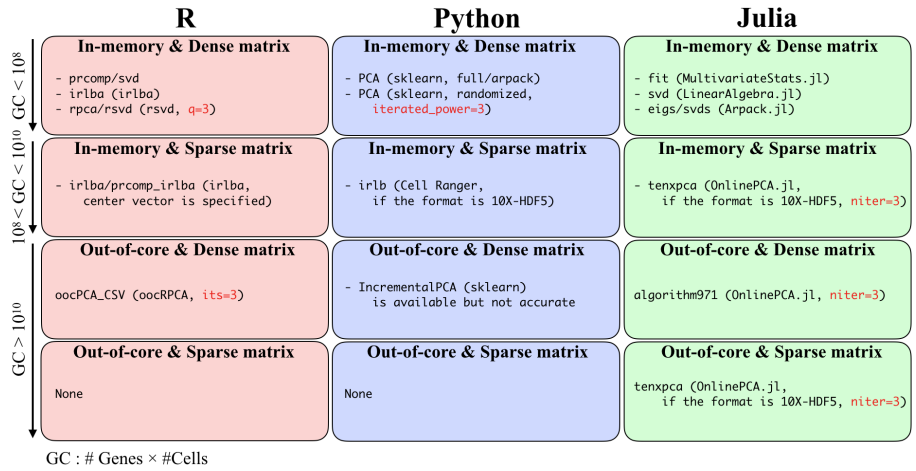

Figure 14 User guidelines. Recommended PCA implementations categorized based on written language and matrix size.

bioRxiv preprint doi: https://doi.org/10.1101/642595; this version posted May 20, 2019. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

Tables

Table 1 Use cases of PCA implementations in scRNA-seq studies.

scRNA-seq studies	PCA implementations	Commands or functions used in the studies
In most cases		`prcomp`/`svd` (R)
[13, 40, 41, 49, 50, 53, 54, 56, 64, 67, 69, 72, 74, 76, 79, 85, 87]	Golub-Kahan method	`PCA` (Python, *sklearn*)
Bhaduri *et al.*, [88]	Downsampling	Unknown
Loompy [87]	SKL	`IncrementalPCA` (Python, *sklearn*)
	IRLBA	`PCA` (Python, *sklearn*)
Scanpy [87]	SKL	`IncrementalPCA` (Python, *sklearn*)
	Halko's method	`TruncatedSVD` (Python, *sklearn*)
Cell Ranger [22]	IRLBA	`irlb` (Python, from scratch)
Seurat2 [47]	IRLBA	`irlba` (R, *irlba*)
Scran [48]	Golub-Kahan method	`svd` (R)
	IRLBA	`irlba` (R, *irlba*)
SAFE [66]	IRLBA	`irlba` (R, *irlba*)
	Golub-Kahan method	`svds` (MATLAB)
MAGIC [50]	Halko's method	`randPCA` (MATLAB, from scratch)
	Halko's method	`PCA` (Python, *sklearn*)
Harmony [55]	IRLBA	`irlba` (R, *irlba*)
Scater [76]	Golub-Kahan method	`prcomp` (R)
	IRLBA	`irlba` (R, *irlba*)
GiniClust2 [68]	IRLBA	`propack.svd` (R, *svd*)
SIMLR [65]	Halko's method	`fast.rsvd` (R, from scratch)
SEQC [83]	Golub-Kahan method	`PCA` (Python, *sklearn*)
	Halko's method	`PCA` (Python, *sklearn*)
CellFishing.jl [70]	algorithm971	`rsvd` (Julia, from scratch)

Table 2 Real-world datasets for benchmarking

Dataset	Total variance	# Genes	# Cells	# Cell types	PCs used	File size (Normalized, CSV)	File size (Count, CSV)	File size (Count, Binary)
Cortex	4170.7	21614	1679	21	PC1-10	650 MB	113 MB	17 MB
Pancreas	254.9	17499	8569	14	PC1-12	2.7 GB	287 MB	23 MB
Brain	205.0	23771	1306127	60	PC1-20	316 GB	58 GB	3.2 GB

Additional Files

Additional File 1 — Pseudo-code of all the PCA algorithms. (PDF 178 KB)
Additional File 2 — Source code of all the PCA implementations. (PDF 58 KB)
Additional File 3 — Pair plots of all the PCA (Cortex) implementations. (ZIP 7.2 MB)
Additional File 4 — Pair plots of all the PCA (Pancreas) implementations. (ZIP 6.7 MB)
Additional File 5 — Pair plots of all the PCA (Brain) implementations. (ZIP 11.6 MB)
Additional File 6 — Crashed jobs caused by out-of-memory errors. (TXT 930 B)
Additional File 7 — Parameter tuning of the orthogonal iteration, gradient descent, and stochastic gradient descent implementations. (PDF 1.3 MB)
Additional File 8 — Parameter tuning of the randomized SVD implementations. (PDF 988 KB)
Additional File 9 — Developer guidelines. (PNG 1.1 MB)