Open access • Posted Content • DOI:10.1101/727867

Analysis of single-cell RNA sequencing data based on autoencoders

Andrea Tangherloni, Federico Ricciuti, Daniela Besozzi, Pietro Liò ...+3 more authors

Institutions: Wellcome Trust, University of Milano-Bicocca, University of Cambridge, Wellcome Trust Sanger Institute

Published on: 29 Mar 2021 - bioRxiv (Cold Spring Harbor Laboratory)

BMC Bioinformatics

RESEARCH

Open Access

Analysis of single-cell RNA sequencing data based on autoencoders

Andrea Tangherloni[1,2,3,4*], Federico Ricciuti[5], Daniela Besozzi[5,6], Pietro Liò[7*†] and Ana Cvejic[1,2,3*†]

*Correspondence: andrea.tangherloni@unibg.it; pl219@cam.ac.uk;as889@cam.ac.uk
†Pietro Liò and Ana Cvejic contributed equally to this work.
[1] Wellcome Trust-Medical Research Council Cambridge Stem Cell Institute, Cambridge CB2 0AW, UK[4] Present Address: Department of Human and Social Sciences, University of Bergamo, 24129 Bergamo, Italy
[7] Department of Computer Science and Technology, University of Cambridge, Cambridge CB3 0FD, UK
Full list of author information is available at the end of the article

Abstract

Background: Single-cell RNA sequencing (scRNA-Seq) experiments are gaining ground to study the molecular processes that drive normal development as well as the onset of different pathologies. Finding an effective and efficient low-dimensional representation of the data is one of the most important steps in the downstream analysis of scRNA-Seq data, as it could provide a better identification of known or putatively novel cell-types. Another step that still poses a challenge is the integration of different scRNA-Seq datasets. Though standard computational pipelines to gain knowledge from scRNA-Seq data exist, a further improvement could be achieved by means of machine learning approaches.

Results: Autoencoders (AEs) have been effectively used to capture the non-linearities among gene interactions of scRNA-Seq data, so that the deployment of AE-based tools might represent the way forward in this context. We introduce here scAEspy, a unifying tool that embodies: (1) four of the most advanced AEs, (2) two novel AEs that we developed on purpose, (3) different loss functions. We show that scAEspy can be coupled with various batch-effect removal tools to integrate data by different scRNA-Seq platforms, in order to better identify the cell-types. We benchmarked scAEspy against the most used batch-effect removal tools, showing that our AE-based strategies outperform the existing solutions.

Conclusions: scAEspy is a user-friendly tool that enables using the most recent and promising AEs to analyse scRNA-Seq data by only setting up two user-defined parameters. Thanks to its modularity, scAEspy can be easily extended to accommodate new AEs to further improve the downstream analysis of scRNA-Seq data. Considering the relevant results we achieved, scAEspy can be considered as a starting point to build a more comprehensive toolkit designed to integrate multi single-cell omics.

Keywords: Autoencoders, scRNA-Seq, Dimensionality reduction, Clustering, Batch correction, Data integration

Background

Single-cell RNA sequencing (scRNA-Seq) was named the "Method of the Year" in 2013, and it is currently used to investigate cell-to-cell heterogeneity since it allows for measuring the transcriptome-wide gene expression at single-cell resolution, enabling the identification of different cell-types. scRNA-Seq data are prevalent generated in studies

that aim at understanding the molecular processes driving normal development and the onset of pathologies [1, 2]. This field of research continuously poses new computational questions that have to be addressed [3].

One of the most important steps in scRNA-Seq analysis is the clustering of cells into groups that correspond to known or putatively novel cell-types, by considering the expression of common sets of signature genes. However, this step still remains a challenging task because applying clustering approaches in high-dimensional spaces can generate misleading results, as the distance between most pairs of points is similar [4]. As a consequence, finding an effective and efficient low-dimensional representation of the data is one of the most crucial steps in the downstream analysis of scRNA-Seq data. A common workflow of downstream analysis, depicted in Fig. 1, includes two dimensionality reduction steps: (1) Principal Component Analysis (PCA) [5] for an initial reduction of the dimensions based on the Highly Variable Genes (HVGs), and (2) a non-linear dimensionality reduction approach—e.g., t-distributed Stochastic Neighbour Embedding (t-SNE) [6] or Uniform Manifold Approximation and Projection (UMAP) [7]—on the PCA space for visualisation purposes (e.g., showing the labelled clusters) [8, 9]. In addition, when multiple scRNA-Seq datasets have to be combined for further analyses, the technical non-negligible batch-effects that may exist among the datasets must be taken into account [3, 8, 10–13], making the dimensionality reduction even

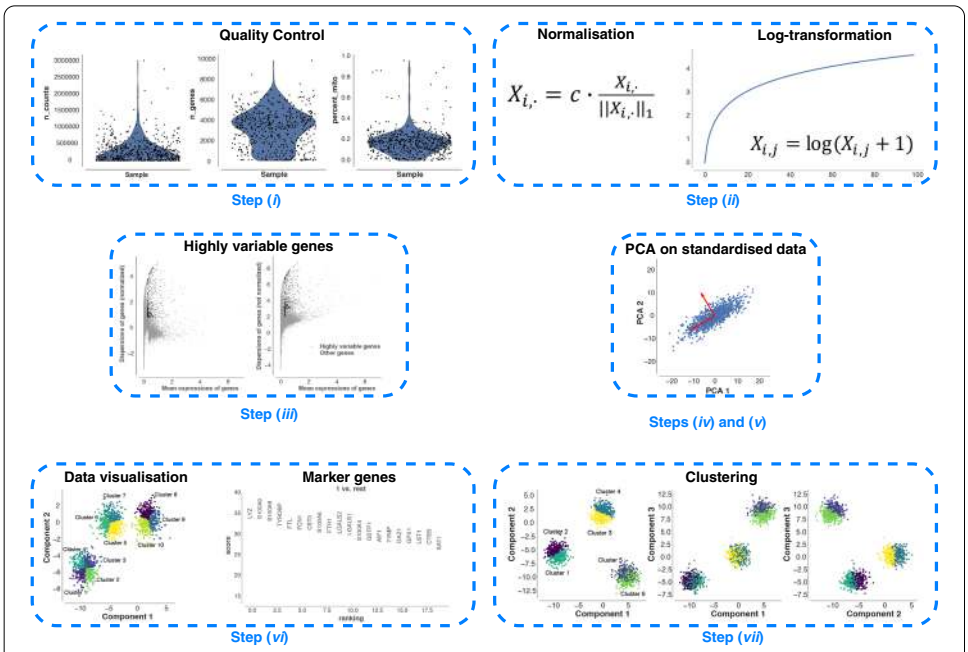


Fig. 1 A common workflow for the downstream analysis of scRNA-Seq data. The workflow includes the following seven steps: (*i*) quality control to remove low-quality cells that may add technical noise, which could obscure the real biological signals; (*ii*) normalisation and log-transformation; (*iii*) identification of the HVGs to reduce the dimensionality of the dataset by including only the most informative genes; (*iv*) standardisation of each gene to zero mean and unit variance; (*v*) dimensionality reduction generally obtained by applying PCA; (*vi*) clustering of the cells starting from the low-dimensional representation of the data that are used to annotate the obtained clusters (i.e., identification of known and putatively novel cell-types); (*vii*) data visualisation on the low-dimensional space generated by applying a non-linear approach (e.g., t-SNE or UMAP) on the reduced space calculated in step (*v*)

more complicated and fundamental. Indeed, finding a salient batch corrected and a low dimensional embedding space can help to better partition and distinguish the various cell-types.

Although commonly used approaches for dimensionality reduction achieved good performance when applied to scRNA-Seq data [8], novel and more robust dimensionality reduction strategies should be used to account for the sparsity, intrinsic noise, unexpected dropout, and burst effects [3, 14], as well as the low amounts of RNA that are typically present in single-cells. Ding *et al.* showed that low-dimensional representations of the original data learned using latent variable models preserve both the local and global neighbour structures of the original data [15]. Autoencoders (AEs) showed outstanding performance in this regard due to their ability to capture the strong non-linearities among the gene interactions existing in the high-dimensional expression space.

Autoencoders for denoising and dimensionality reduction

Deep Count AE network (DCA) was one of the first AE-based approach proposed to denoise scRNA-Seq datasets [16] by considering the count distribution, overdispersion, and sparsity of the data. DCA relies on a negative binomial noise model, with or without zero-inflation, to capture non-linear gene-gene dependencies. Starting from the vanilla version of the Variational AE (VAE) [17], several approaches have been proposed. Among them, single-cell Variational Inference (scVI) was the first scalable framework that allowed for a probabilistic representation and analysis of gene expression datasets [18]. scVI was built upon Deep Neural Networks (DNNs) and stochastic optimization to consider the information across similar cells and genes to approximate the distributions underlying the analysed gene expression data. This computational tool allows for coupling low-dimensional probabilistic representation of gene expression data with the downstream analysis to consider the measurement of uncertainty through a statistical model. Svensson *et al.* integrated a Linearly Decoded VAE (LDVAE) into scVI [19], enabling the identification of relationships among the cell representation coordinates and gene weights via a factor mode. Single-cell VAE (scVAE) was introduced to directly model the raw counts from RNA-seq data [20]. More importantly, the authors proposed a Gaussian-mixture model to better learn biologically plausible groupings of scRNA-Seq data on the latent space.

The framework called Decomposition using Hierarchical AE (scDHA) is composed of two modules [21]. The first module is a non-negative kernel AE able to provide a non-negative, part-based denoised representation of the original data. During this step, the genes and the components having an insignificant contribution to the denoised representation of the data are removed. The second module is a stacked Bayesian self-learning network built upon the VAE. This specific module is used to project the denoised data into a low-dimensional space used during the downstream analysis. scDHA outperformed PCA, t-SNE, and UMAP in terms of silhouette index [22] on the tested datasets.

AEs coupled with disentanglement methods have been used to both improve the data representation and obtain better separation of the biological factors of variation in gene expression data [23]. In addition, a graph AE, consisting of graph convolutional layers, was developed to predict relationships between single-cells. This framework can be used to identify the cell-types in the dataset under analysis and discover the driver genes

for the differentiation process. Wang *et al.* proposed a deep VAE for scRNA-Seq data named VASC [24], a deep multi-layer generative model that improves the dimensionality reduction and visualisation steps in an unsupervised manner. Thanks to its ability to model dropout events—that can hinder various downstream analysis steps (e.g., clustering analysis, differential expression analysis, inference of gene-to-gene relationships) by introducing a high number of zero counts in the expression matrices—and to find non-linear hierarchical representations of the data, VASC obtained superior performance with respect to four state-of-the-art dimensionality reduction and visualisation approaches [24].

Dimensionality Reduction with Adversarial VAE (DR-A) has been recently proposed to fulfil the dimensionality reduction step from a data-driven point of view [25]. Compared to the previous approaches, DR-A exploits an adversarial VAE-based framework, which is a recent variant of generative adversarial networks. DR-A generally obtained more accurate low-dimensional representation of scRNA-Seq data compared to state-of-the-art approaches (e.g., PCA, scVI, t-SNE, UMAP), leading to better clustering performance. Geddes *et al.* proposed an AE-based cluster ensemble framework to improve the clustering process [26]. As a first step, random subspace projections of the data are compressed onto a low-dimensional space by exploiting an AE, obtaining different encoded spaces. Then, an ensemble clustering approach is applied across all the encoded spaces to generate a more accurate clustering of the cells.

Autoencoders for the imputation of missing data

AutoImpute was proposed to deal with the insufficient quantities of starting RNA in the individual cells, a problem that generally leads to significant dropout events. As a consequence, the resulting gene expression matrices are sparse and contain a high number of zero counts. AutoImpute is an AE-based imputation method that works on sparse gene expression matrices, trying to learn the inherent distribution of the input data to assign the missing values [27]. scSVA was also proposed to identify and recover dropout events [28], which are imputed by fitting a mixed model of each possible cell-type. In addition, it performs an efficient feature extraction step of the high-dimensional scRNA-Seq data, obtaining a low-dimensional embedding. In the tests showed by the authors, scSVA was able to outperform different state-of-the-art and novel approaches (e.g., PCA, t-SNE, UMAP, VASC).

Other two methods based on non-parametric AEs were proposed to address the imputation problem [29]. Learning with AuToEncoder (LATE) relies on an AE that is directly trained on a gene expression matrix with parameters randomly generated, while TRANSfer learning with LATE (TRANSLATE) takes into consideration a reference gene expression dataset to estimate the parameters that are then used by LATE on the new gene expression matrix. LATE and TRANSLATE were able to obtain outstanding performance on both real and simulated data by recovering non-linear relationships in pairs of genes, allowing for a better identification and separation of the cell-types.

GraphSCI combines Graph convolution network and AE, and it is the first approach that systematically integrates gene-to-gene relationships with the gene expression data into a deep learning framework. GraphSCI is able to impute the dropout events

by taking advantage of low-dimensional representations of similar cells and gene-gene interactions [30].

Generally, in the existing AEs the input data are usually codified in a specific format, making their integration into the existing scRNA-Seq analysis toolkits (e.g., Scanpy [31] and Seurat [32]) a difficult task. In addition, the existing tools are implemented in Keras (https://github.com/fchollet/keras), TensorFlow [33], or PyTorch [34], and all the three libraries are thus required to run them. Finally, the currently available AEs cannot be directly exploited to obtain the latent space or to generate synthetic cells. In order to overcome the described limitations, we developed scAEspy, which is a unifying, user-friendly, and standalone tool that relies only on TensorFlow and allows easy access to different AEs by setting up only two user-defined parameters. scAEspy can be used on High-Performance Computing (HPC) infrastructures to speed-up its execution. It can be easily run on clusters of both Central Processing Units (CPUs) and Graphics Processing Units (GPUs). Indeed, it was designed and developed to be executed on multi- and many-core infrastructures. In addition, scAEspy gives access to the latent space, generated by the trained AE, which can be directly used to show the cells in this embedded space or as a starting point for other dimensionality reduction approaches (e.g., t-SNE and UMAP) as well as downstream analyses (e.g., batch-effect removal).

In this work, we show how scAEspy can be used to deal with the existing batch-effects among samples. Indeed, the application of batch-effect removal tools into the latent space (Fig. 2) allowed us to outperform state-of-the-art methods as well as the same batch-effect removal tools applied on the PCA space. Finally, scAEspy implements different loss functions, which are fundamental to deal with different sequencing platforms.

Results

We tested PCA and AEs to address the integration of different datasets. Specifically, we used all the AEs implemented in our scAEspy tool: VAE [17], an AE only based on the Maximum Mean Discrepancy (MMD) distance (called here MMDAE) [35], MMD-VAE, Gaussian-mixture VAE (GMVAE), and two novel Gaussian-mixture AEs that we developed, called GMMMD and GMMMDVAE, respectively. In all the performed tests, the constrained versions of the following loss functions were used: Negative Binomial (NB), Poisson, zero-inflated NB (ZINB), zero-inflated Poisson (ZIP). We used a number of Gaussian distributions equal to the number of datasets to integrate for GMVAE, GMMMD, and GMMMDVAE. In addition, we tested the following configurations of the hidden layer and latent space to understand how the dimension of the AEs might potentially affect their performance: (256, 64), (256, 32), (256, 16), (128, 64), (128, 32), (128, 16), (64, 32), and (64, 16), where (H, L) represents the number of neurons composing the hidden layer (H neurons) and latent space (L neurons).

In order to deal with the possible batch-effects, we applied the following approaches, as suggested in [8, 11] and being the most used batch-effect removal tools in the literature: Batch Balanced k-Nearest Neighbours (BBKNN) [36], Harmony [37], ComBat [38–40], and the Seurat implementation of the Canonical Correlation Analysis (CCA) [13]. Thus, we compared vanilla PCA and AEs, PCA and AEs followed by either BBKNN or Harmony (Fig. 2), ComBat, and CCA. For each batch-effect removal tool, we used the different samples that have to be merged as batches.

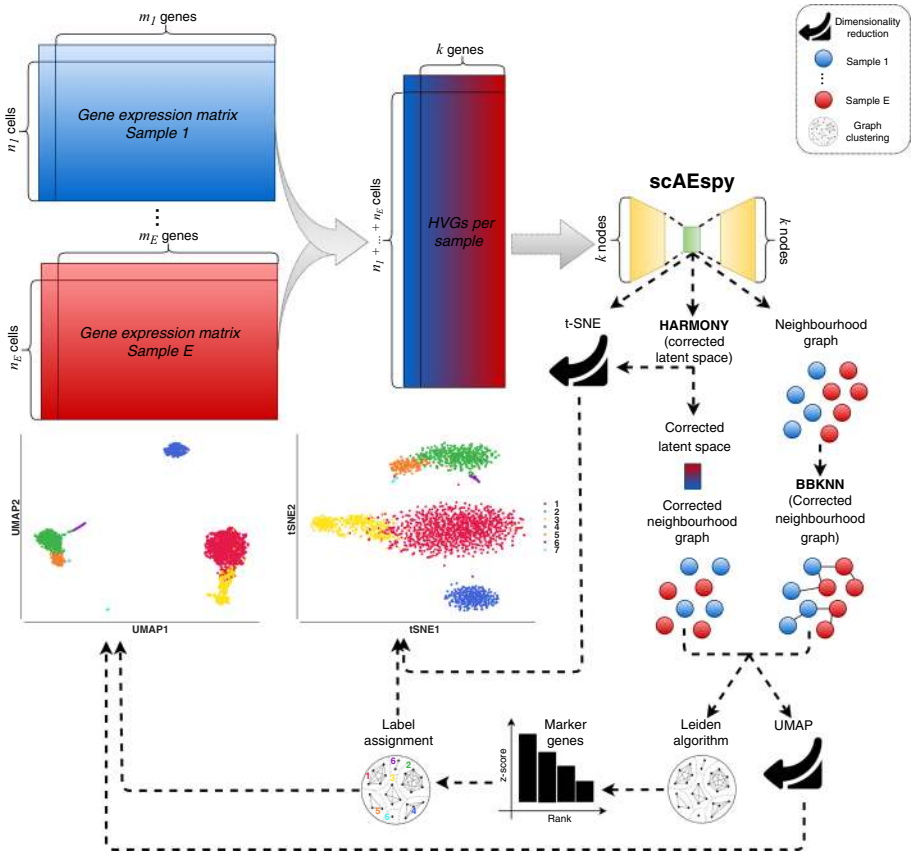


Fig. 2 The proposed workflow to integrate different samples. Given E different samples, their gene expression matrices are merged. Then, the top k HVGs are selected by considering the different samples. Specifically, they are selected within each sample separately and then merged to avoid the selection of batch-specific genes. scAEspy is used to reduce the HVG space (k dimensions), and the obtained latent space can be (*i*) used to calculate a t-SNE space, (*ii*) corrected by Harmony, and (*iii*) used to infer an uncorrected neighbourhood graph. The corrected latent space by Harmony is then used to build a neighbourhood graph, which is clustered by using the Leiden algorithm and used to calculate a UMAP space. Otherwise, BBKNN is applied to rebuild a uncorrected neighbourhood graph by taking into account the possible batch-effects. The corrected neighbourhood graph built by BBKNN is then clustered by using the Leiden algorithm and used to calculate a UMAP space. In order to assign the correct label to the obtained clusters, the marker genes are calculated by using the Mann–Whitney U test. Finally, the annotated clusters can be visualised in both t-SNE and UMAP space

The proposed strategies were compared on three publicly available datasets, namely: Peripheral Blood Mononuclear Cells (PBMCs), Pancreatic Islet Cells (PICs), and Mouse Cell Atlas (MCA) by using well-known clustering metrics (i.e., Adjusted Rand Index, Adjusted Mutual Information Index, Fowlkes Mallows Index, Homogeneity Score, and V-Measure). It is worth mentioning that generally the cell-types are manually identified by expert biologists starting from an overclustering or underclustering of the data, possibly followed by different steps of subclustering of some clusters. Here, we evaluate how the different strategies are able to automatically separate the cells by fixing the number of clusters equal to the number of cell-types manually identified by the authors of the papers.

Datasets

Peripheral blood mononuclear cells

PBMCs from eight patients with systemic lupus erythematosus were collected and processed using the 10× Chromium Genomics platform [41]. The dataset is composed of a control group (6573 cells) and an interferon-β stimulated group (7466 cells). We considered the 8 distinct cell-types identified by the authors following a standard workflow [41]. The count matrices were downloaded from Seurat's tutorial *"Integrating stimulated vs. control PBMC datasets to learn cell-type specific responses"* (https://satijalab.org/seurat/v3.0/immune_alignment.html).

Pancreatic islet cells

PIC datasets were generated independently using four different platforms: CEL-Seq [42] (1004 cells), CEL-Seq2 [43] (2285 cells), Fluidigm C1 [44] (638 cells), and Smart-Seq2 [45] (2394 cells). For our tests, we considered the 13 different cell-types across the datasets identified in [46] by applying PCA on the scaled integrated data matrix. The count matrices were downloaded from Seurat's tutorial *"Integration and Label Transfer"* (https://satijalab.org/seurat/v3.0/integration.html).

Mouse cell atlas

MCA is composed of two different datasets. The former was generated by Han *et al.* [47] using Microwell-Seq (4239 cells) [47], while the latter by the Tabula Muris Consortium [48] using Smart-Seq2 (2715 cells). The 11 distinct cell-types with the highest number of cells, which were present in both datasets, have been taken into account as in [11]. The count matrices were downloaded from the public GitHub repository related to [11] (https://github.com/JinmiaoChenLab/Batch-effect-removal-benchmarking).

Metrics

Adjusted rand index

The Rand Index (RI) is a similarity measure between the results obtained from the application of two different clustering methods. The first clustering method is used as ground truth (i.e., true clusters), while the second one has to be evaluated (i.e., predicted clusters). The RI is calculated by considering all pairs of samples appearing in the clusters, namely, it counts the pairs that are assigned either to the same or different clusters in both the predicted and the true clusters. The Adjusted RI (ARI) [49] is the "adjusted for chance" version of the RI. Its values vary in the range $[-1, 1]$: a value close to 0 means a random assignment, independently of the number of clusters, while a value equal to 1 indicates that the clusters obtained with both clustering approaches are identical. Negative values are obtained if the index is less than the expected index.

Adjusted mutual information index

The Mutual Information Index (MII) [50] represents the mutual information of two random variables, which is a similarity measure of the mutual dependence between the two variables. Specifically, it is used to quantify the amount of information that can be gained by one random variable observing the other variable. The MII is strictly correlated with the entropy of a random variable, which quantifies the expected "amount

of information" that is contained in a random variable. This index is used to measure the similarity between two labels of the same data. Similarly to ARI, the Adjusted MII (AMII) is "adjusted for chance" and its values vary in the range [0, 1].

Fowlkes mallows index

The Fowlkes Mallows Index (FMI) [51] measures the similarity between the clusters obtained by using two different clustering approaches. It is defined as the geometric mean between precision and recall. Assuming that the first clustering approach is the ground truth, the precision is the percentage of the results that are relevant, while the recall refers to the percentage of total relevant results correctly assigned by the second clustering approach. The index ranges from 0 to 1.

Homogeneity score

The result of the tested clustering approach satisfies the Homogeneity Score (HS) [52] if all of its clusters contain only cells that belong to a single cell-type. Its values range from 0 to 1, where 1 indicates perfectly homogeneous labelling. Notice that by switching true cluster labels with the predicted cluster labels, the Completeness Score is obtained.

Completeness score

The result of the tested clustering approach satisfies the Completeness Score (CS) [52] if all the cells that belong to a given cell-type are elements of the same cluster. Its values range from 0 to 1, where 1 indicates perfectly complete labelling. Notice that by switching true cluster labels with the predicted cluster labels, the HS is obtained.

V-measure

The V-Measure (VM) [53] is the harmonic mean between HS and CS; it is equivalent to MII when the arithmetic mean is used as aggregation function.

Integration of multiple datasets obtained with the same sequencing platforms

Nowadays, various scRNA-Seq platforms are currently available (e.g., droplet-based and plate-based [54–63]) and their integration is often challenging due to the differences in biological sample batches as well as to the used experimental platforms. To test whether AEs can be effectively applied to combine multiple datasets, generated using the same platform but under different experimental conditions, we used the PBMC datasets.

We merged the control and treated datasets by first using vanilla PCA and AEs, and then PCA and AEs followed by either BBKNN or Harmony, ComBat, and CCA. After the construction of the neighbourhood graphs, we performed a clustering step by using the Leiden algorithm [64]. Since in the original paper 8 different cell-types were manually identified [41], we selected Leiden's resolutions that allowed us to obtain 8 distinct clusters and calculated all the metrics described above. In what follows, the calculated values of all metrics are given in percentages (mean $\pm$ standard deviation). For each metric, the higher the value the better the result.

Our analysis showed that the CCA-based approach, proposed in the Seurat library, achieved a mean ARI equal to 73.49% ($\pm$1.52%), ComBat reached a mean ARI of 72.84% ($\pm$0.78%), vanilla PCA had a mean ARI of 68.90% ($\pm$0.86%), PCA followed

by BBKNN was able to obtain a mean ARI of 83.65% (±0.81%), while PCA followed by Harmony reached a mean ARI of 82.83% (±1.20%), as shown in Fig. 3a. Among all the tested AEs, MMDAE followed by Harmony (using the NB loss function, 256 neurons for the hidden layer, and 32 neurons for the latent space) achieved the best results, with a mean ARI equal to 87.18% (±0.49%). In order to assess whether any of the results obtained by the best AE were different from a statistical point of view, we applied the Mann–Whitney U test with the Bonferroni correction [65–67]. In all the comparisons, MMDAE followed by Harmony had a p-value lower than 0.0001, confirming that the achieved results are statistically different compared to those achieved by the other approaches.

Regarding the AMII, CCA had a mean value of 66.46% (±0.50%), ComBat achieved a mean value of 70.95% (±0.82%), vanilla PCA obtained a mean value of 68.44% (±1.00%), PCA followed by BBKNN reached a mean value of 75.22% (±0.76%), while PCA followed by Harmony reached a mean value of 74.55% (±1.19%). MMDAE followed by Harmony had better results, with a mean value equal to 78.61% (±0.29%). MMDAE followed by Harmony outperformed the other strategies also in terms of of FMS, HS, CS, and VM (see Additional file 2 and Fig. 4).

We also compared the results obtained by the best AE for each of the tested dimension (H, L) in terms of ARI (Fig. 3b). GMMMD followed by Harmony (using the NB loss function) obtained the best results for the dimension (64, 16), GMMMD followed by Harmony (using the Poisson loss function) reached the best results for the dimensions (128, 16) and (256, 64), and GMMMD followed by BBKNN (using the NB loss function) achieved the best results for the dimension (128, 32). MMDAE followed by Harmony (using the NB loss function) was able to reach the best results for the dimensions (64, 32), (256, 16), and (256, 32), while MMDAE followed by Harmony (using the Poisson loss function) obtained the best result for the dimensions (128, 64). Notice that we used two Gaussian distributions because we merged two different datasets.

In order to visually assess the quality of the separation of the manually annotated cell-type and the found clusters, we plotted them in the UMAP space generated starting from the MMDAE followed by Harmony space (Fig. 3c, d). Finally, we also plotted the two samples in the same UMAP space to visually see the quality of the alignment between the two samples themselves (Fig. 5a). This plot confirms that the batch-effects were completely removed.

Our analysis showed that clustering the neighbourhood graph generated from AE spaces allowed for a better identification of the existing cell-types when compared to other approaches, thus confirming the ARI results.

Integration of multiple datasets obtained with different sequencing platforms

Combining datasets from different studies and scRNA-Seq platforms can be a powerful approach to obtain complete information about the biological system under investigation. However, when datasets generated with different platforms are combined, the high variability in the gene expression matrices can obscure the existing biological relationships. For example, the gene expression values are much higher in data acquired with plate-based methods (i.e., up to millions) than in those acquired with droplet-based

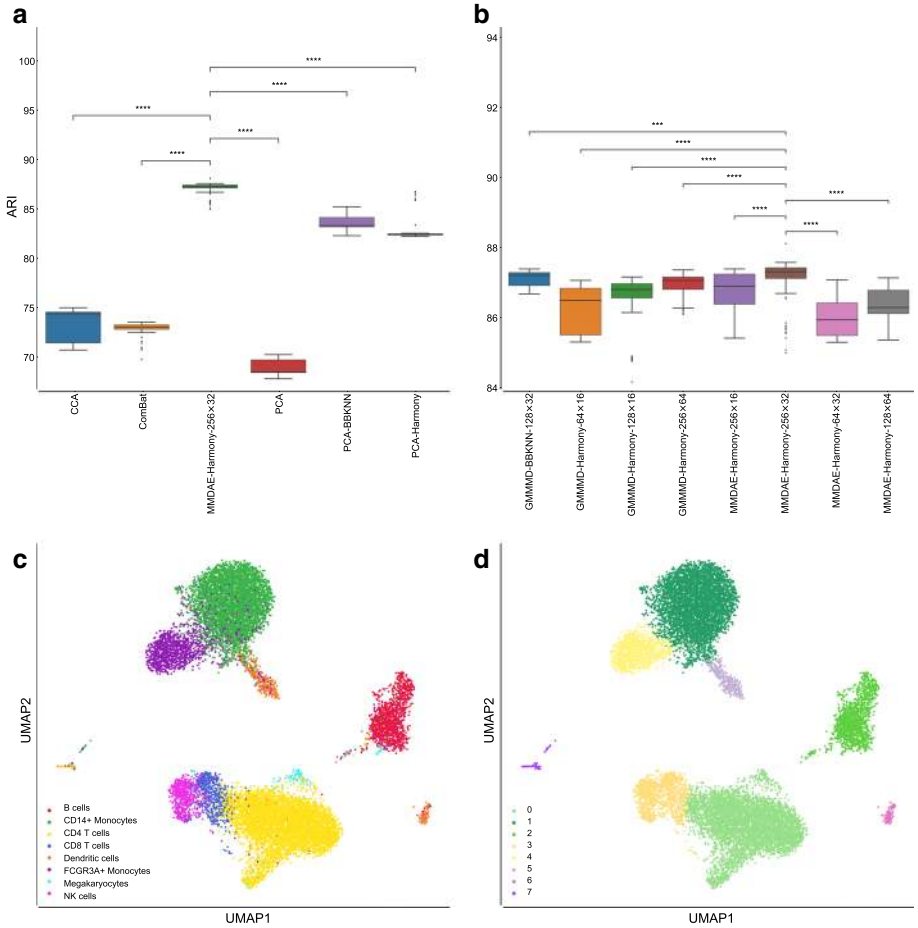


Fig. 3 Results obtained on the PBMC datasets. **a** Boxplot showing the ARI values achieved by CCA, ComBat, PCA, MMDAE followed by Harmony with dimension (256, 32), PCA followed by BBKNN, and PCA followed by Harmony on the PBMC datasets. **b** Boxplot showing the ARI values achieved by the best AE for each of the tested dimension (*H*, *L*) of the hidden layer (*H* neurons) and latent space (*L* neurons). **c** UMAP visualisation of the cell-type manually annotated in the original paper. **d** UMAP visualisation of clusters identified by the Leiden algorithm using the resolution corresponding by the best ARI achieved by MMDAE followed by Harmony. p-value $\leq$ 0.0001 (****); 0.0001 $<$ p-value $\leq$ 0.001 (***); 0.001 $<$ p-value $\leq$ 0.01 (**); 0.01 $<$ p-value $\leq$ 0.05 (*); p-value $>$ 0.05 (ns)

methods (i.e., a few thousands). Thus, combining gene expression data that spread across several orders of magnitude is a difficult task that cannot be tackled by using linear approaches like PCA. To examine how well AEs perform in solving this issue, we combined four PIC datasets acquired with CEL-Seq [56], CEL-Seq2 [57], Fluidigm C1 [63], and Smart-Seq2 protocols [62].

We integrated the datasets by first using vanilla PCA and AEs, and then PCA and AEs followed by either BBKNN or Harmony, ComBat, and CCA. Since in the original paper 13 cell-types were manually annotated for the PIC datasets [46], we clustered the neighbourhood graphs using the Leiden algorithm considering only the resolutions that allowed us to obtain 13 distinct clusters. We then calculated ARI, AMII, FMS, HS, CS, and VM metrics. The calculated values of all metrics are given

in percentages (mean ± standard deviation); for each metric, the higher the value the better the result.

CCA had a very low mean ARI, i.e., 5.45% (±0.22%), ComBat obtained a mean ARI of 76.20% (±3.06%), vanilla PCA achieved a mean ARI of 61.38% (±0.09%), PCA followed by BBKNN reached a mean ARI of 71.49% (±0.58%), while PCA followed by Harmony was able to obtain a mean ARI of 94.00% (±0.36%), see Fig. 6a. GMMMD followed by Harmony (using the NB loss function, 256 neurons for the hidden layer, and 32 neurons for the latent space) outperformed the other AEs, achieving a mean ARI equal to 94.23% (±0.12%). In all the comparisons, except for the one against PCA followed by Harmony, GMMMD followed by Harmony had a p-value lower than 0.0001, confirming that the achieved results are statistically different with respect to those obtained by the other approaches.

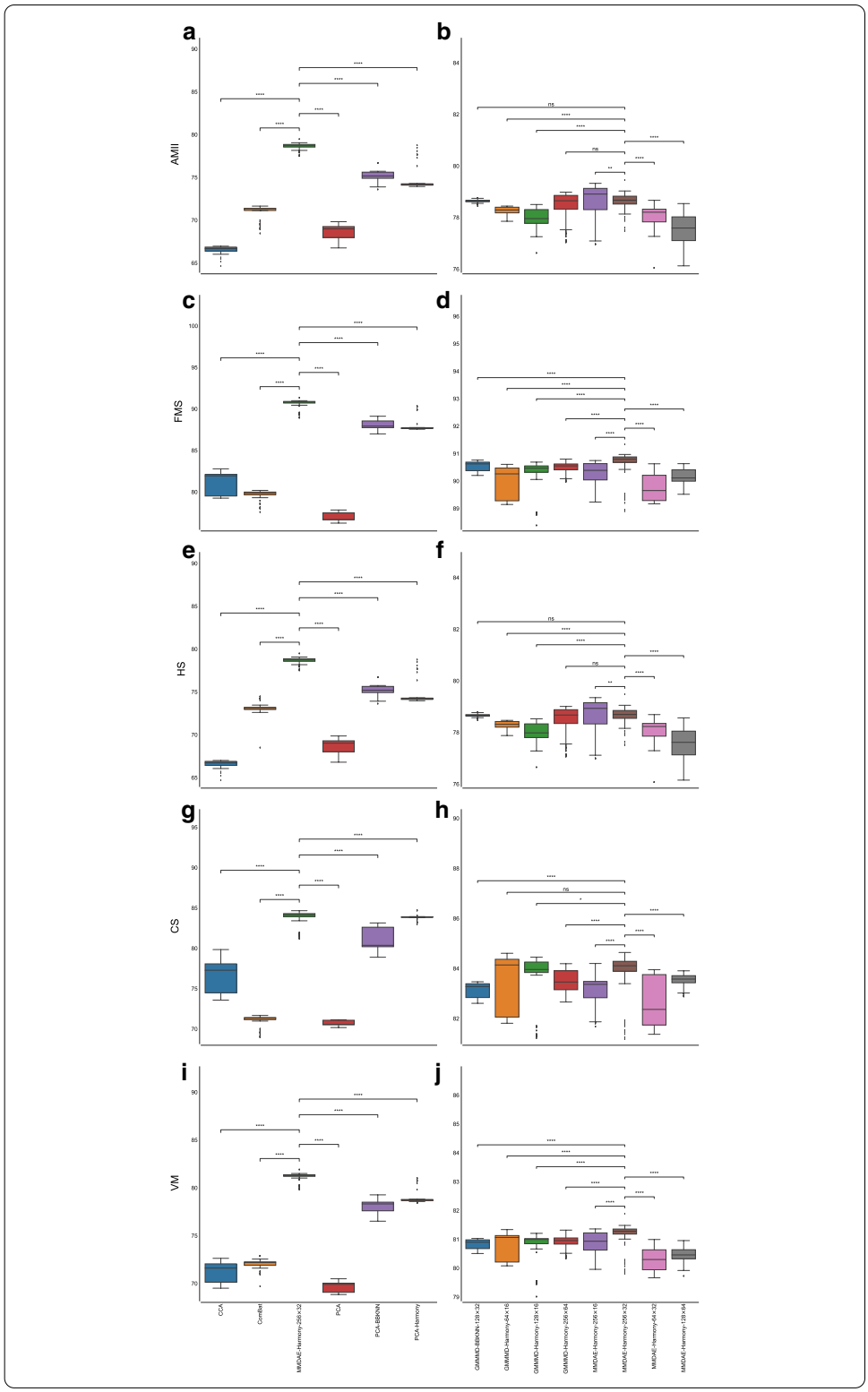
Similar results were achieved for the AMII metric: CCA reached a mean value equal to 16.57% (±0.33%), ComBat obtained a mean value of 76.11% (±1.14%), vanilla PCA reached a mean value of 71.70% (±0.11%), PCA followed by BBKNN achieved a mean value of 77.55% (±0.65%) and PCA followed by Harmony a mean value of 91.17% (±0.47%), while GMMMD followed by Harmony was able to reach a mean value equal to 89.37% (±0.02%). Considering the other measures, both PCA and GMMMD followed by Harmony obtained very similar results, outperforming the other strategies (see Additional file 3 and Fig. 7).

Considering the best AE for each of the tested dimension (H, L) in terms of ARI (see Fig. 6b), GMMMD followed by Harmony (using the NB loss function) resulted the best choice for the dimensions (128, 64) and (256, 32), while it obtained the best results for the dimension (64, 32) when the Poisson loss function was used. GMVAE followed by Harmony (using the Poisson loss function) reached the best results for the dimensions (128, 32) and (256, 16). MMDAE followed by Harmony achieved the best results for the dimensions (256, 64) and (64, 16), exploiting the NB loss function and Poisson loss function, respectively. Finally, VAE followed by Harmony obtained the best results with the Poisson function for the dimension (128, 16). Note that we exploited four Gaussian distributions because we merged four different datasets.

The quality of the separation of the manually annotated cell-type and found clusters can be visually evaluated in Fig. 6c, d. We finally visualised the cells (coloured by platform) using the UMAP space generated from the GMMMD followed by Harmony space (Fig. 5b) to confirm that the batch-effects among the samples sequenced with different platforms were correctly removed. Taken together, our analysis shows that GMMMD followed by Harmony can efficiently identify the "shared" cell-types

(See figure on next page.)

Fig. 4 Boxplot showing the values of the calculated metrics using CCA, ComBat, PCA, MMDAE followed by Harmony with dimension (256, 32), PCA followed by BBKNN, and PCA followed by Harmony as well as by the best AE for each of the tested dimension (H, L), analysing the PBMC datasets. **a** AMII achieved by the different strategies. **b** AMII achieved by the best AE for each of the tested dimension. **c** FMS achieved by the different strategies. **d** FMS achieved by the best AE for each of the tested dimension. **e** HS achieved by the different strategies. **f** HS achieved by the best AE for each of the tested dimension. **g** CS achieved by the different strategies. **h** CS achieved by the best AE for each of the tested dimension. **i** VM achieved by the different strategies. **j** VM achieved by the best AE for each of the tested dimension. p-value $\leq$ 0.0001(****); 0.0001 $<$ p-value $\leq$ 0.001(***); 0.001 $<$ p-value $\leq$ 0.01(**); 0.01 $<$ p-value $\leq$ 0.05 (*); p-value $>$ 0.05 (ns)


a
b
c
d
e
f
g
h
i
j
AMI
FMS
HS
CS
VM
CCA
Combat
MMDAE-Harmony-256×32
PCA
PCA-BBKNN
PCA-harmony
GMMMD-BBKNN-128×32
GMMMD-Harmony-64×16
GMMMD-Harmony-128×16
GMMMD-Harmony-256×64
MMDAE-Harmony-256×16
MMDAE-Harmony-256×32
MMDAE-Harmony-64×32
MMDAE-Harmony-128×64

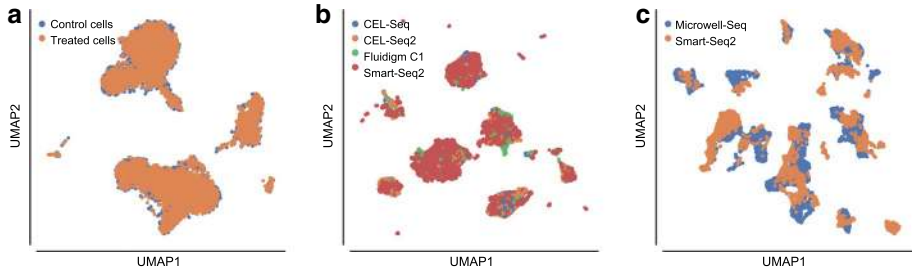


Fig. 5 UMAP visualisation showing the sample alignment performed by Harmony into the latent space obtained by MMDAE with dimension (256, 32) for the PBMC datasets **a**, by GMMMD with dimension (256, 32) for the PIC datasets **b**, and by MMDVAE with dimension (256, 64) for the MCA datasets **c**

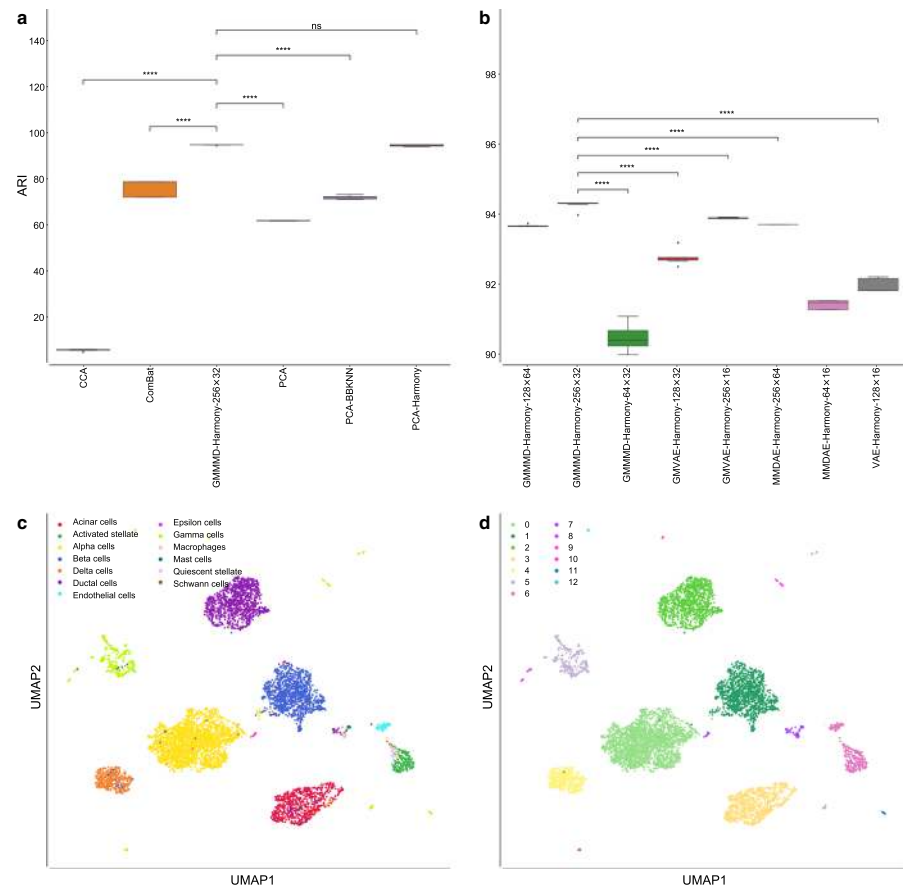


Fig. 6 Results obtained on the PIC datasets. **a** Boxplot showing the ARI values achieved by CCA, ComBat, PCA, GMMMD followed by Harmony with dimension (256, 32), PCA followed by BBKNN, and PCA followed by Harmony on the PBMC datasets. **b** Boxplot showing the ARI values achieved by the best AE for each of the tested dimension (*H*, *L*) of the hidden layer (*H* neurons) and latent space (*L* neurons). **c** UMAP visualisation of the cell-type manually annotated in the original paper. **d** UMAP visualisation of clusters identified by the Leiden algorithm using the resolution corresponding by the best ARI achieved by GMMMD followed by Harmony. p-value ≤ 0.0001 (****); $0.0001 < p$-value ≤ 0.001 (***); $0.001 < p$-value ≤ 0.01 (**); $0.01 < p$-value ≤ 0.05 (*); p-value > 0.05 (ns)

across the different platforms due to its ability to deal with the high variability in the gene expression matrices. We would like to highlight that PCA followed by Harmony was capable of achieving good results because the original clusters were obtained by applying a similar pipeline [46].

As a final test, we combined the two MCA datasets acquired with Microwell-Seq [47] and Smart-Seq2 protocols [62]. We integrated the datasets in the same way we did in the other two tests. We clustered the neighbourhood graphs using the Leiden algorithm considering only the resolutions that allowed us to obtain 11 distinct clusters because 11 distinct cell-types were manually annotated for the PIC datasets [47]. We then calculated all metrics.

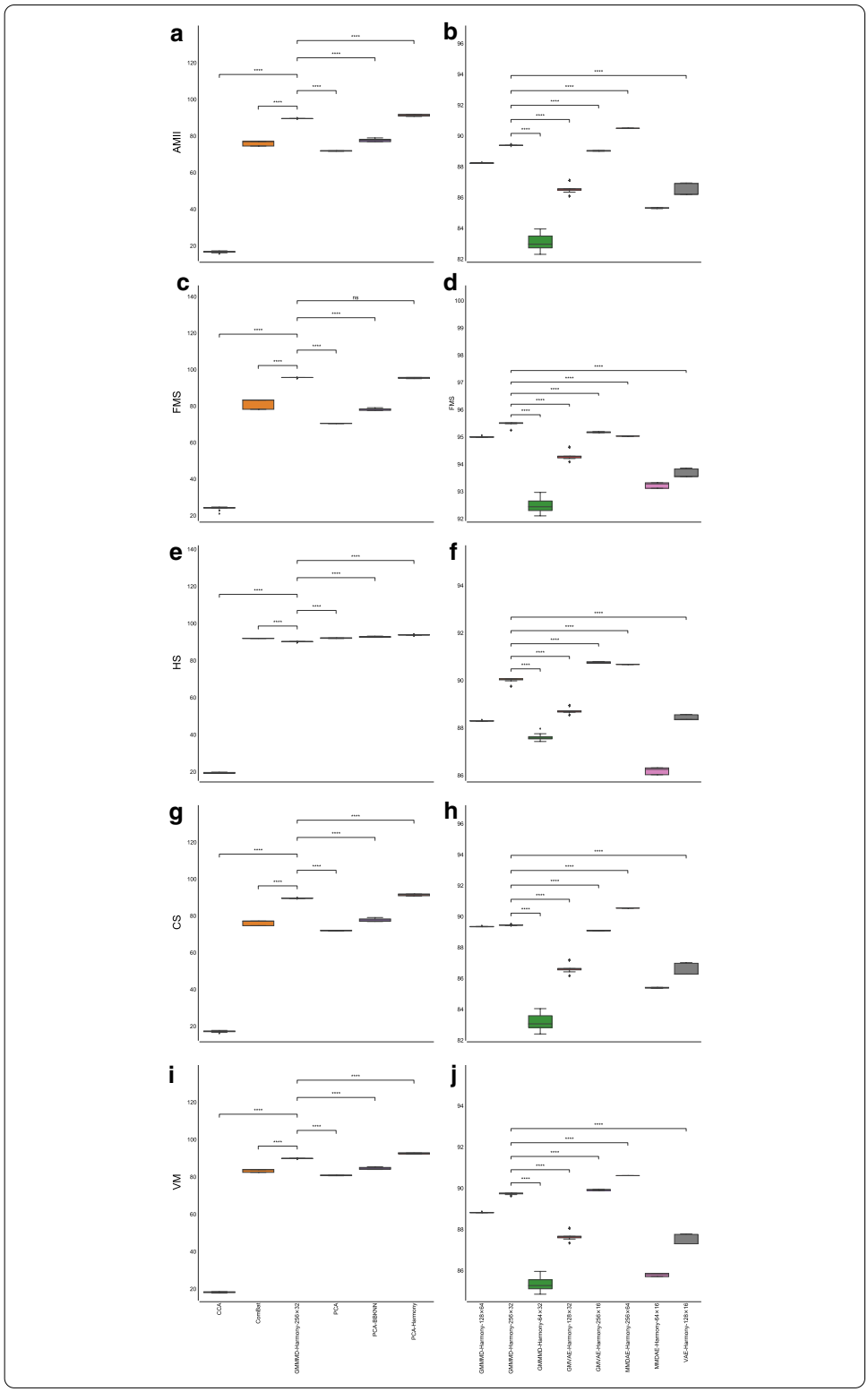
In such a case, MMDVAE followed by Harmony (using the Poisson loss function, 256 neurons for the hidden layer, and 64 neurons for the latent space) outperformed the other AEs as well as the other strategies, obtaining a mean ARI equal to 79.50% (±0.02%), as shown in Fig. 8a. ComBat achieved the worst mean ARI, i.e., 54.13% (±4.22%), CCA reached a mean ARI of 57.62% (±0.75%), vanilla PCA obtained a mean ARI of 67.29% (±11.55%), PCA followed by BBKNN had a similar mean ARI, that is, 67.73% (±3.98%), while PCA followed by Harmony achieved a mean ARI of 66.08% (±0.11%). MMDVAE followed by Harmony had a p-value lower than 0.0001 in all the tested comparisons. Considering the other metrics, MMDVAE followed by Harmony generally obtained better results compared to the other strategies (see Additional file 4, Figs. 8b, and 9).

Comparing the best AE for each of the tested dimension (H, L) in terms of ARI, the vanilla GMMMD with the NB loss function obtained the best results for the dimension (128, 16), while GMMMD followed by Harmony reached the best results for the dimensions (256, 32) and (64, 16), exploiting the Poisson loss function and NB loss function, respectively. MMDAE followed by BBKNN (using the ZIP loss function) achieved the best results for the dimensions (256, 16) and (128, 64), exploiting the NB loss function and Poisson loss function, respectively. MMDVAE followed by Harmony resulted the best choice for the dimensions (128, 16) and (256, 64) when coupled with the NB loss function and Poisson loss function, respectively. Finally, VAE followed by Harmony with the Poisson loss function obtained the best results for the dimension (64, 32). As for the the integration of the PBMC datasets, we used two Gaussian distributions because we merged two different datasets.

Figure 8c, d show the UMAP generated from the MMDVAE followed by Harmony space coloured by the manually annotated cell-type and found clusters, respectively, while Fig. 5c depicts the cells coloured by platform on the same UMAP space,

(See figure on next page.)

Fig. 7 Boxplot showing the values of the calculated metrics using CCA, ComBat, PCA, GMMMD followed by Harmony with dimension (256, 32), PCA followed by BBKNN, and PCA followed by Harmony as well as by the best AE for each of the tested dimension (H, L), analysing the PIC datasets. **a** AMII achieved by the different strategies. **b** AMII achieved by the best AE for each of the tested dimension. **c** FMS achieved by the different strategies. **d** FMS achieved by the best AE for each of the tested dimension. **e** HS achieved by the different strategies. **f** HS achieved by the best AE for each of the tested dimension. **g** CS achieved by the different strategies. **h** CS achieved by the best AE for each of the tested dimension. **i** VM achieved by the different strategies. **j** VM achieved by the best AE for each of the tested dimension. p-value $\leq$ 0.0001(****); 0.0001 $<$ p-value $\leq$ 0.001(***); 0.001 $<$ p-value $\leq$ 0.01(**); 0.01 $<$ p-value $\leq$ 0.05 (*); p-value $>$ 0.05 (ns)


a
b
c
d
e
f
g
h
i
j
AMII
FMS
HS
CS
VM
CCA
ComBat
GMMMD-Harmony-256×32
PCA
PCA-BBKNN
PCA-Harmony
GMMMD-Harmony-128×64
GMMMD-Harmony-256×32
GMMMD-Harmony-64×32
GMVAE-Harmony-128×32
GMVAE-Harmony-256×16
MMDAE-Harmony-256×64
MMDAE-Harmony-64×16
VAE-Harmony-128×16

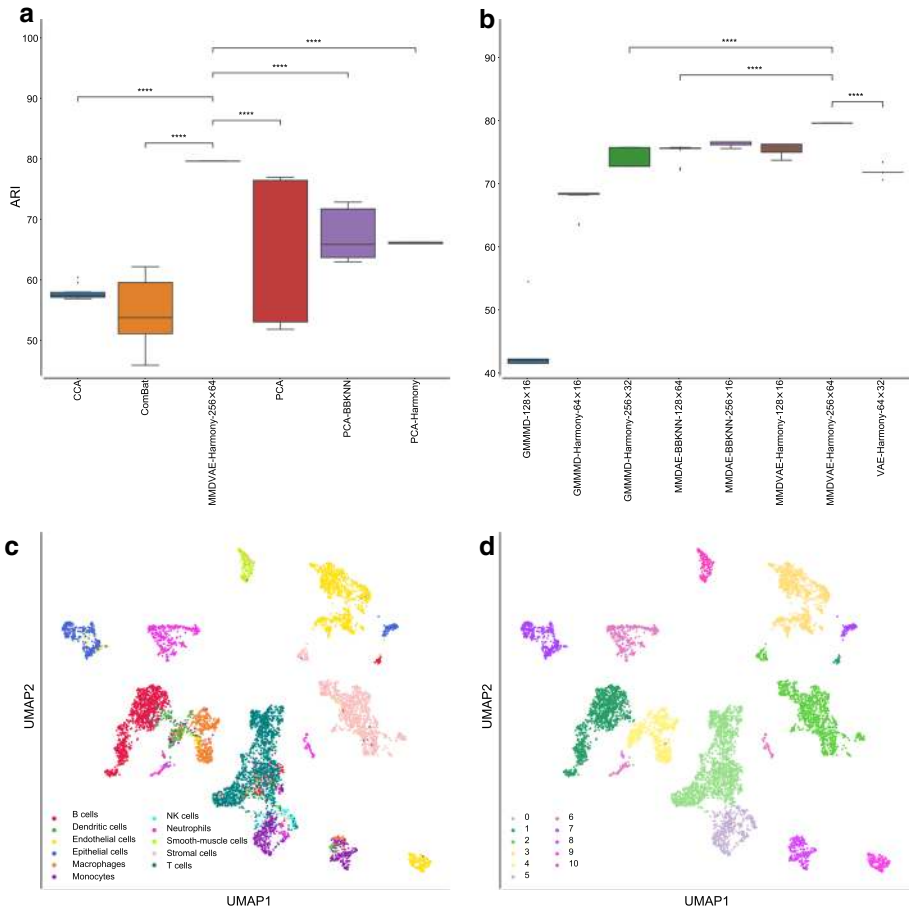


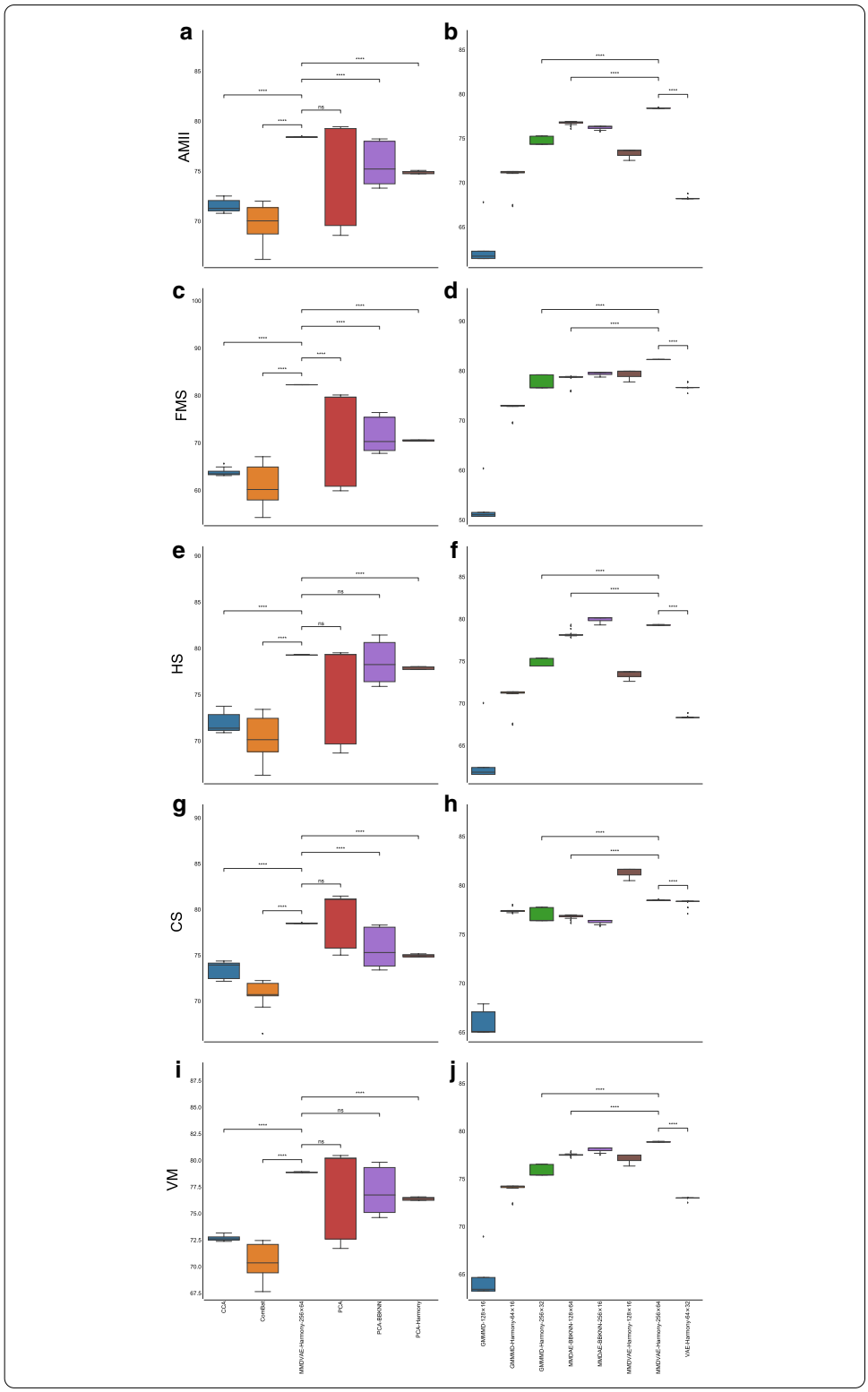
Fig. 8 Results obtained on the MCA datasets. **a** Boxplot showing the ARI values achieved by CCA, ComBat, PCA, MMDVAE followed by Harmony with dimension (256, 64), PCA followed by BBKNN, and PCA followed by Harmony. **b** Boxplot showing the ARI values achieved by the best AE for each of the tested dimension (*H*, *L*) of the hidden layer (*H* neurons) and latent space (*L* neurons). **c** UMAP visualisation of the cell-type manually annotated in the original paper. **d** UMAP visualisation of clusters identified by the Leiden algorithm using the resolution corresponding by the best ARI achieved by MMDVAE followed by Harmony. p-value $\leq$ 0.0001(****); 0.0001 $<$ p-value $\leq$ 0.001(***); 0.001 $<$ p-value $\leq$ 0.01(**); 0.01 $<$ p-value $\leq$ 0.05 (*); p-value $>$ 0.05 (ns)

confirming that the batch-effects between the two samples were correctly removed. In this case, the achieved results show that MMDVAE followed by Harmony was able to better identify the "shared" cell-types across the different platforms.

In order to test the computational performance of scAEspy, we considered a VAE with a single hidden layer composed of 256 neurons, a latent space of 32 neurons, and

(See figure on next page.)

Fig. 9 Boxplot showing the values of the calculated metrics using CCA, ComBat, PCA, MMDVAE followed by Harmony with dimension (256, 64), PCA followed by BBKNN, and PCA followed by Harmony, as well as by the best AE for each of the tested dimension (*H*, *L*), analysing the MCA datasets. **a** AMII achieved by the different strategies. **b** AMII achieved by the best AE for each of the tested dimension. **c** FMS achieved by the different strategies. **d** FMS achieved by the best AE for each of the tested dimension. **e** HS achieved by the different strategies. **f** HS achieved by the best AE for each of the tested dimension. **g** CS achieved by the different strategies. **h** CS achieved by the best AE for each of the tested dimension. **i** VM achieved by the different strategies. **j** VM achieved by the best AE for each of the tested dimension. p-value $\leq$ 0.0001(****); 0.0001 $<$ p-value $\leq$ 0.001(***); 0.001 $<$ p-value $\leq$ 0.01(**); 0.01 $<$ p-value $\leq$ 0.05 (*); p-value $>$ 0.05 (ns)


a
b
c
d
e
f
g
h
i
j
AMII
FMS
HS
CS
VM
CCA
Combat
MMD-VAE-Harmony-256-x64
PCA
PCA-BBKNN
PCA-Harmony
GMMD-128-x16
GMMD-Harmony-x64-x16
GMMD-Harmony-256-x32
MMDAE-BBKNN-128-x64
MMDAE-BBKNN-256-x16
MMD-VAE-Harmony-128-x16
MMD-VAE-Harmony-256-x64
VAE-Harmony-64-x32

the NB loss function. All tests were executed on a laptop equipped with an Intel Core i7-1165G7 CPU (clock 2.8 GHz) and 16 GB of RAM, running on Ubuntu 20.04 LTS. In all the tested datasets, we used the top 1000 HVGs. The executions lasted 95.02 seconds, 101.05 seconds, and 223.45 seconds for PIC, MCA, and PBMC, respectively. Considering that PIC, MCA, and PBMC are composed of 6321, 6954, and 14039 cells, respectively, scAEspy scales linearly with the number of cells.

Discussion

Non-linear approaches for dimensionality reduction can be effectively used to capture the non-linearities among gene interactions that may exist in the high-dimensional expression space of scRNA-Seq data [15]. Among the different non-linear approaches, AEs showed outstanding performance, outperforming other approaches like UMAP and t-SNE. Several AE-based methods have been developed so far, but their integration with the common single-cell toolkits results in a difficult task because they usually require input data codified in a specific format. In addition, three different machine learning libraries are required to run them (i.e., Keras, TensorFlow, PyTorch).

Here, we proposed scAEspy, a unifying and user-friendly tool that allows the user to use the most recent and promising AEs (i.e., VAE, MMDAE, MMDVAE, and GMVAE). We also designed and developed GMMMD and GMMMDVAE, two novel AEs that combine MMDAE and MMDVAE with GMVAE to exploit more than one Gaussian distribution. We introduced a learnable prior distribution in the latent space to model the dimensionality of the subpopulations of cells composing the data, or to combine multiple samples.

We integrated AEs with both Harmony and BBKNN to remove the existing batch-effects among different datasets. Our results showed that exploiting the latent space to remove the existing batch-effects permits for better identification of the cell subpopulations. As a batch-effect removal tool, Harmony allowed for achieving better results than BBKNN in the majority of the cases. When different droplet-based data have to be combined, our GMMMD and the MMDAE, coupled with the constrained NB and Poisson loss functions, obtained the best results compared to all the other AEs. In order to combine and analyse multiple datasets, generated by using different scRNA-Seq platforms, both GMMMD and MMDVAE, mainly used together with the NB and Poisson loss functions, outperformed the other strategies. However, also GMVAE and the simple VAE obtained outstanding performance, highlighting that the Kullback-Leibler divergence function can become fundamental to handle data spreading various orders of magnitude, especially the high values (up to millions) introduced by plate-based methods. It is clear that using more than one Gaussian distribution allows for obtaining a better integration of the datasets and separation of the cell-types when more than two datasets have to be integrated, as clearly shown by the results reached on the PIC datasets.

As a good practice, we suggest filtering the gene expression matrices using the top HVGs (i.e., 1000), calculated separately within each batch and merged to avoid the selection of batch-specific genes. The original counts, corresponding to the top HVGs, should be the input data of scAEspy. GMMMD coupled with Harmony should be used to integrate different droplet-based datasets as well as to merge datasets generated by using

different scRNA-Seq platforms. Harmony should be applied to the latent space produced by GMMMD to take into account the possible batches among the datasets. When a single dataset has to be analysed, the users should use MMDAE with droplet-based data, while MMDVAE should be preferred for data acquired with plate-based methods. In all cases, using a single hidden layer composed of 256 neurons, a latent space with 32 neurons, and the constrained NB or Poisson loss function generally allows for obtaining satisfactory results.

Considering the achieved results on the identification of the clusters, scAEspy can be used at the basis of methods that aim at automatically identifying the cell-types composing the scRNA-Seq datasets under analysis [68]. As a matter of fact, scAEspy coupled with BBKNN was successfully applied to integrate 15 different foetal human samples, enabling the identification of rare blood progenitor cells [69].

Conclusions

In this study, we proposed an AE-based and user-friendly tool, named scAEspy, which allows for using the most recent and promising AEs to analyse scRNA-Seq data. The user can select the desired AE by only setting up two user-defined parameters. Once the selected AE has been trained, it can be used to generate synthetic cells to increase the number of data for further downstream analyses (e.g., training classifiers). In scAEspy, the latent space is easily accessible and thus allows the user to perform different types of analysis, such as the correction of possible batch-effect in a reduced non-linear space or the inference of differentiation trajectories. In this case, the latent space can be utilised to generate the "pseudotime" that measures transcriptional changes that a cell undergoes during the dynamic process.

Thanks to its modularity, scAEspy can be extended to accommodate new AEs so that the user will be always able to utilise the latest and cutting-edge AEs [70], which can improve the downstream analysis of scRNA-Seq data. It is worth noticing that scAEspy can be used on HPC infrastructures, both based on CPUs and GPUs, to speed-up the computations. This is a crucial point when datasets composed of hundreds of thousands of cells are analysed. In such cases, the required running time drastically increases, so relying on HPC infrastructures is the best solution to incredibly reduce the prohibitive running time.

Future improvements

As an improvement, prior biological knowledge about genes from ontologies can be incorporated into scAEspy. Ontologies can introduce useful information into machine learning systems that are used to solve biological problems. They allow for integrating data from different omics (e.g., genomics, transcriptomics, proteomics, and metabolomics) as structured representations of semantic knowledge, which is commonly used for the representation of biological concepts. This approach has been successfully applied to predict the clinical targets from high-dimensional low-sample data [71]. Specifically, ontology embeddings are able to capture the semantic similarities among the genes, which can be exploited to sparsify the network connections. In addition, the Gene Ontology (GO) [72] can be exploited to interpret the extracted features from the latent spaces generated by the AEs, thus bringing

an explanation to the learned representations of the gene expression data. As a possible example, g:Profiler [73] focusing on GO terms, Kyoto Encyclopedia of Genes and Genomes (KEGG), and Reactome can be used on the learned embeddings to investigate the joint effects of different gene sets within specific biological pathways. This approach can help the interpretability and explainability of the learned embeddings of the used AEs.

Integration of multi-omics data

Since AEs showed outstanding performance in the integration of multi-omics of cancer data [70], we plan to extend scAEspy to analyse other single-cell omics. For instance, AEs can be applied to analyse scATAC-seq, where the identification of the cell-types is still very difficult due to technical challenges [10, 74]. scAEspy could be effectively applied to analyse disparate types of single-cell data from different points of view. The latent representations of different or combined single-cell omics can be used for further and more in-depth analyses. For instance, the application of other machine learning techniques (e.g., deep neural networks) to the latent representations could facilitate the identification of interesting patterns on gene expression or methylation data, as well as relationships among genomics variants. In that regard, scAEspy can be the starting point to build a more comprehensive toolkit designed to integrate multi single-cell omics as an integration and extension of the work proposed in [70].

Methods

We developed scAEspy so that it can be easily integrated into both Scanpy and Seurat pipelines, as it directly works on a gene expression matrix (see Fig. 2). We integrated into a single tool the latest and most powerful AEs designed to resolve the problems underlying scRNA-Seq data (e.g., sparsity, intrinsic noise, dropout events [3]). Specifically, scAEspy is comprised of six AEs, based on the VAE [17] and InfoVAE [35] architectures. The following most advanced AEs are included in scAEspy: VAE, MMDAE, MMDVAE, GMVAE, and two novel Gaussian-mixture AEs that we developed, called GMMMD and GMMMDVAE. GMMMD is a modification of the MMDAE where more than one Gaussian distribution is used to model different modes and only the MMD function is used as divergence function. GMMMDVAE is a combination of MMDVAE and GMVAE where both the MMD function [75] and the Kullback-Leibler divergence function [76] are used. scAEspy allows the user to exploit these six different AEs by setting up two user-defined parameters, α and λ, which are needed to balance the MMD and the Kullback-Leibler divergence functions. We designed and developed GMMMD and GMMMDVAE starting from InfoVAE [35] and scVAE [20]. In addition, a learnable mixture distribution was used for the prior distribution in the latent space, and also the marginal conditional distribution was defined to be a learnable mixture distribution with the same number of components as the prior distribution. Finally, the user can also select the following loss functions: NB, constrained NB, Poisson, constrained Poisson, ZINB, constrained ZINB, ZIP, constrained ZIP, and Mean Square Error (MSE).

The tested batch-effect removal tools

Originally proposed to deal with batch-effects in microarray gene expression data [38], ComBat has been successfully applied to analyse scRNA-Seq data [77]. Briefly, given

a gene expression matrix, it is firstly standardised so that all genes have similar means and variances. Then, starting from the obtained standardised matrix, standard distributions are fitted using a Bayesian approach to estimate the existing batch-effects in the data. Finally, the original expression matrix is corrected using the computed batch-effect estimators. In our tests, we used the default parameter settings provided by the Scanpy function `combat`. We then applied PCA on the space obtained by the top k (here, we set $k = 1000$) HVGs calculated by using the function provided by Scanpy (v.1.4.5.1), where the top HVGs are separately selected within each batch and merged to avoid the selection of batch-specific genes. We calculated the first 50 components and applied the so-called "elbow method" to select the number of components for the downstream analysis [8]. The "elbow method" was applied taking into consideration the variance explained by each PCA component, which can be visualised using the Scanpy function `pca_variance_ratio`. Including less informative PCA components might help in the identification of rare cell types, but they can also introduce noise that hinders the downstream analysis steps; thus, we opted to include only the most informative PCA components. Specifically, we used the first 18, 13, and 18 components for PBMC, PIC, and MCA datasets, respectively. After that, we calculated the neighbourhood graph by using the default parameter settings proposed in Scanpy. We clustered the obtained neighbourhood graphs with the Leiden algorithm by selecting the values of the resolution parameter such that the number of clusters was equal to the manually annotated clusters. Finally, all the metrics for each found resolution have been calculated.

As another batch-effect removal tool, we used the CCA-based approach proposed in the Seurat package (v.2.3.4) [13]. We applied both `RunCCA` and `MultiCCA` Seurat functions to integrate two batches and more than two batches, respectively. Firstly, we normalised and log-transformed the counts. Then, we calculated the top 1000 HVGs by using the function provided by Scanpy (v.1.4.5.1). We also scaled the log-transformed data to zero mean and unit variance. In both `RunCCA` and `MultiCCA` Seurat functions, as a first step, the CCA components were exploited to compute the linear combinations of the genes with the maximum correlation between the batches. We used the default number of CCA components (i.e., 20) provided by the functions `RunCCA` and `MultiCCA` of the Seurat package, as also suggested in [36]. A dynamic time warping (`AlignSubspace` Seurat function), which accounts for population density changes, was then used to align the calculated vectors and obtain a single low-dimensional subspace where the batch-effects are corrected. We calculated the neighbourhood graph, using the default parameter settings proposed in Scanpy, starting from the aligned low-dimensional subspace. We clustered the built neighbourhood graphs with the Leiden algorithm as explained before. Finally, we calculated all the metrics for each found resolution.

We also applied Harmony [37] to remove the batch-effects. Starting from a reduced space (e.g., PCA space or latent space), Harmony exploits an iterative clustering-based procedure to remove the multiple-dataset-specific batch-effects. In each iteration, the following four steps are applied: (*i*) the cells are grouped into multiple-dataset clusters by exploiting a variant of the soft k-means clustering, which is a fast and flexible method developed to cluster single-cell data; (*ii*) a centroid is calculated for each cluster and for each specific dataset; (*iii*) using the calculated centroids, a correction factor is derived for each dataset; (*iv*) the correction factors are then used to correct each cell with a cell-specific factor.

As a further batch-effect removal tool, we applied BBKNN [36]. Polanski *et al.* [36] showed that BBKNN has comparable or better performance in removing batch-effects with respect to the CCA-based approach proposed in the Seurat package, Scanorama [78], and mnnCorrect [79]. In addition, BBKNN is a lightweight graph alignment method that requires minimal changes to the classical workflow. Indeed, it computes the k-nearest neighbours in a reduced space (e.g., PCA or latent space), where the nearest neighbours are identified in a batch-balanced manner using a user-defined distance (in our tests, we used the Euclidean distance). The neighbour information is transformed into connectivities to build a graph where all cells across batches are linked together. We used both Harmony and BBKNN to correct the PCA and AE spaces.

As a final step, we calculated the UMAP spaces starting from the built neighbourhood graphs and using the default parameter settings proposed in Scanpy, except for the initialisation of the low dimensional embedding (i.e., *init_pos* equal to *random*, and *random_state* equal to 10 of the `umap` function).

The proposed pipeline

We modified the workflow shown in Fig. 1 by replacing PCA with AEs (Fig. 2). We merged the gene expression matrices of E different samples ($E = 2$, $E = 4$, and $E = 2$ for PBMC, PIC, and MCA datasets, respectively). We applied both PCA and AEs on the space obtained by the top 1000 HVGs calculated by using the implementation provided by Scanpy (v.1.4.5.1).

For what concerns PCA, we firstly normalised and log-transformed the counts, then we applied a classic standardisation, that is, the distribution of the expression of each gene was scaled to zero mean and unit variance. We calculated the first 50 components; after that, we used the "elbow method" to select the first 12, 14, and 19 components for the PBMC, PIC, and MCA datasets, respectively. As in the case of the ComBat workflow, the "elbow method" was applied considering the variance explained by each PCA component.

Regarding AEs, we used the original counts since AEs showed to achieve better results when applied using the raw counts [20]. Indeed, using the counts allows for exploiting discrete probability distributions, such as Poisson and NB distributions, which obtained the best results in our tests. In all the tests presented here, we used a single hidden layer. In addition, we set 100 epochs, sigmoid activation functions, and a batch equal to 100 samples (i.e., cells). In all tests we used the Adam optimizer [80].

After that, we applied three different strategies (Fig. 2): (*i*) we calculated the neighbourhood graph in both PCA and AE spaces by using the default parameter settings proposed in Scanpy. Then, we clustered the obtained neighbourhood graphs with the Leiden algorithm as described before. Finally, we calculated all the metrics for each found resolution; (*ii*) we performed a similar analysis where we firstly corrected the PCA and AE spaces using Harmony [37] with the default parameter settings proposed in https://github.com/slowkow/harmonypy; (*iii*) we performed the same analysis described in (*i*) by replacing the neighbourhood graphs with those generated using BBKNN, using the default parameter settings.

Table 1 Setting of α, λ, and K to obtain the desired AE

	α	λ	K
VAE	0	1	1
MMDAE	1	1	1
MMDVAE	0	2	1
GMVAE	0	1	> 1
GMMMD	1	1	> 1
GMMMDVAE	0	2	> 1

It is worth mentioning that in all the tested batch-effect removal tools, we used the different samples that have to be merged as batches. Specifically, the number of batches is equal to 2, 4, and 2 for the PBMC, PIC, and MCA datasets, respectively.

The generalised formulation of scAEspy

In this work, we used the notation proposed in [35] to extend MMDVAE with multiple Gaussian distributions as well as to introduce a learnable prior distribution in the latent space. The idea behind the introduction of learnable coefficients is that they might be suitable to model the diversity among the subpopulations of cells composing the data or to combine multiple samples or datasets.

We consider $p^*(\mathbf{x})$ as the unknown probability in the input space over which the optimisation problem is formulated, $\boldsymbol{z}$ is the latent representation of $\boldsymbol{x}$ with $|\boldsymbol{z}| \leq |\boldsymbol{x}|$. The encoder is identified by a function $e_{\boldsymbol{\phi}} : \boldsymbol{x} \mapsto \boldsymbol{z}$, while the decoder by a function $d_{\boldsymbol{\theta}} : \boldsymbol{z} \mapsto \boldsymbol{x}$.

We remind that in VAEs, the input $\boldsymbol{x}$ is not mapped into a single point in the latent space, but it is represented by a probability distribution over the latent space. In what follows, we denote by $q(\boldsymbol{z})$ any possible distribution in the latent space and by $y \in \{1, \ldots, K\}$ a categorical random variable, where K corresponds to the number of desired Gaussian distributions. As general strict divergence function, we considered the MMD$(\cdot)$ divergence function [75].

The ELBO term proposed in this work, which is the measure maximised during the training of AEs, is:

$$\begin{aligned} \text{ELBO} = \; & \mathbb{E}[\log(d(\boldsymbol{x}|\boldsymbol{z}, y))] \\ & - (\alpha + \lambda - 1)\text{MMD}(p_e(\boldsymbol{z})||q(\boldsymbol{z})) \\ & - (1 - \alpha)\mathbb{E}[\text{KL}(p_e(\boldsymbol{z}, y|\boldsymbol{x})||q(\boldsymbol{z}, y))], \end{aligned} \tag{1}$$

where KL$(\cdot)$ is the Kullback-Leibler divergence [76] between two distributions. All the mathematical details required to derive the generalised formula shown in Eq. (1) can be found in the Additional file 1.

Equation (1) allows the user to easily exploit VAE, MMDAE, MMDVAE, GMVAE, GMMMD, and GMMMMDVAE (see Table 1).

Availability and requirements

scAEspy is written in Python programming language (v.3.6.5) and it relies on TensorFlow (v.1.12.0), an open-source and massively used machine learning library [33].

scAEspy requires the following Python libraries: NumPy, SciKit-Learn, Matplotlib, and Seaborn. scAEspy's open-source code is available on GitLab: https://gitlab.com/cvejic-group/scaespy under the GPL-3 license. scAEspy can be easily installed using the Python package installer pip, which allows the user to use scAEspy either as a standalone tool exploiting the command line interface, or as a Python package that can be integrated into Python scripts and Jupyter Notebooks.

The repository contains all the scripts, code and Jupyter Notebooks used to obtain the results shown in the paper. In the provided Jupyter Notebooks, we show how easy it is to integrate scAEspy and Scanpy, and how the data can be visualised and explored by using both scAEspy and Scanpy's functions. We also provide a detailed description of scAEspy's parameters so that it can be used by both novice and expert researchers for downstream analyses.

Supplementary information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-021-04150-3.

Additional file 1. Mathematical formulation of the proposed autoencoders. We provide the mathematical derivation of GMMMD and GMMMDVAE as well as the generalised formulation that we derived by following the notation proposed in [35].

Additional file 2. Excel file of the metrics calculated for the PBMC datasets. Each tab is related to a tested approach and shows the calculated metrics and used method.

Additional file 3. Excel file of the metrics calculated for the PIC datasets. Each tab is related to a tested approach and shows the calculated metrics and used method.

Additional file 4. Excel file of the metrics calculated for the MCA datasets. Each tab is related to a tested approach and shows the calculated metrics and used method.

Acknowledgements
We thank Dr. Leonardo Rundo (Department of Radiology, University of Cambridge) for his critical comments.

Author contributions
AT conceived the project. AT and FR developed the software. AC, DB, and PL supervised the project and helped to interpret and present the results. AT performed all the tests and analysed the results. AT wrote the manuscript. AC, DB, and PL edited the manuscript. All authors read and approved the final manuscript.

Funding
This research was supported by Cancer Research UK grant number C45041/A14953 (AC and AT), European Research Council project 677501 – ZF_Blood (AC), Wellcome Trust grant number 203151/Z/16/Z (AC), UK Research and Innovation Medical Research Council grant number MC_PC_17230 (AC), and a core support grant from the Wellcome Trust and MRC to the Wellcome Trust – Medical Research Council Cambridge Stem Cell Institute.

Availability of data and materials
The datasets analysed during the current study as well as all the developed code are available in the scAEspy repository: https://gitlab.com/cvejic-group/scaespy.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Wellcome Trust-Medical Research Council Cambridge Stem Cell Institute, Cambridge CB2 0AW, UK. [2]Department of Haematology, University of Cambridge, Cambridge CB2 0AW, UK. [3]Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Hinxton CB10 1SA, UK. [4]Present Address: Department of Human and Social Sciences, University of Bergamo, 24129 Bergamo, Italy. [5]Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milan, Italy. [6]Bicocca Bioinformatics, Biostatistics and Bioimaging Centre (B4), Milan, Italy. [7]Department of Computer Science and Technology, University of Cambridge, Cambridge CB3 0FD, UK.

Received: 15 February 2021 Accepted: 19 April 2021
Published online: 08 June 2021

References

1. Gladka MM, Molenaar B, De Ruiter H, Van Der Elst S, Tsui H, Versteeg D, Lacraz GP, Huibers MM, Van Oudenaarden A, Van Rooij E. Single-cell sequencing of the healthy and diseased heart reveals cytoskeleton-associated protein 4 as a new modulator of fibroblasts activation. Circulation. 2018;138(2):166–80. https://doi.org/10.1161/CIRCULATIONAHA.117.030742.
2. Keren-Shaul H, Spinrad A, Weiner A, Matcovitch-Natan O, Dvir-Szternfeld R, Ulland TK, David E, Baruch K, Lara-Astaiso D, Toth B, et al. A unique microglia type associated with restricting development of Alzheimer's disease. Cell. 2017;169(7):1276–90. https://doi.org/10.1016/j.cell.2017.05.018.
3. Lähnemann D, Köster J, Szczurek E, McCarthy DJ, Hicks SC, Robinson MD, Vallejos CA, Campbell KR, Beerenwinkel N, Mahfouz A, et al. Eleven grand challenges in single-cell data science. Genome Biol. 2020;21(1):1–35. https://doi.org/10.1186/s13059-020-1926-6.
4. Steinbach M, Ertöz L, Kumar V. The challenges of clustering high dimensional data. In: New directions in statistical physics: econophysics, bioinformatics, and pattern recognition. Berlin: Springer; 2004. p. 273–309. https://doi.org/10.1007/978-3-662-08968-2_16.
5. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemom Intell Lab Syst. 1987;2(1–3):37–52. https://doi.org/10.1016/0169-7439(87)80084-9.
6. Maaten LVD, Hinton G. Visualizing data using t-SNE. J Mach Learn Res. 2008;9:2579–605.
7. Becht E, McInnes L, Healy J, Dutertre C-A, Kwok IW, Ng LG, Ginhoux F, Newell EW. Dimensionality reduction for visualizing single-cell data using UMAP. Nat Biotechnol. 2019;37(1):38. https://doi.org/10.1038/nbt.4314.
8. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol. 2019;15(6):8746. https://doi.org/10.15252/msb.20188746.
9. Hwang B, Lee JH, Bang D. Single-cell RNA sequencing technologies and bioinformatics pipelines. Exp Mol Med. 2018;50(8):96. https://doi.org/10.1038/s12276-018-0071-8.
10. Luecken MD, Buttner M, Chaichoompu K, Danese A, Interlandi M, Müller MF, Strobl DC, Zappia L, Dugas M, Colomé-Tatché M, et al. Benchmarking atlas-level data integration in single-cell genomics. BioRxiv. 2020. https://doi.org/10.1101/2020.05.22.111161.
11. Tran HTN, Ang KS, Chevrier M, Zhang X, Lee NYS, Goh M, Chen J. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. Genome Biol. 2020;21(1):1–32. https://doi.org/10.1186/s13059-019-1850-9.
12. Leek JT, Scharpf RB, Bravo HC, Simcha D, Langmead B, Johnson WE, Geman D, Baggerly K, Irizarry RA. Tackling the widespread and critical impact of batch effects in high-throughput data. Nat Rev Genet. 2010;11(10):733–9. https://doi.org/10.1038/nrg2825.
13. Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol. 2018;36(5):411. https://doi.org/10.1038/nbt.4096.
14. Bacher R, Kendziorski C. Design and computational analysis of single-cell RNA-sequencing experiments. Genome Biol. 2016;17(1):63. https://doi.org/10.1186/s13059-016-0927-y.
15. Ding J, Condon A, Shah SP. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. Nat Commun. 2018;9(1):2002. https://doi.org/10.1038/s41467-018-04368-5.
16. Eraslan G, Simon LM, Mircea M, Mueller NS, Theis FJ. Single-cell RNA-seq denoising using a deep count autoencoder. Nat Commun. 2019;10(1):390. https://doi.org/10.1038/s41467-018-07931-2.
17. Kingma DP, Welling M. Auto-encoding variational bayes. 2013. arXiv preprint arXiv:1312.6114.
18. Lopez R, Regier J, Cole MB, Jordan MI, Yosef N. Deep generative modeling for single-cell transcriptomics. Nat Methods. 2018;15(12):1053. https://doi.org/10.1038/s41592-018-0229-2.
19. Svensson V, Gayoso A, Yosef N, Pachter L. Interpretable factor models of single-cell RNA-seq via variational autoencoders. Bioinformatics. 2020;36(11):3418–21. https://doi.org/10.1093/bioinformatics/btaa169.
20. Grønbech CH, Vording MF, Timshel PN, Sønderby CK, Pers TH, Winther O. scVAE: variational auto-encoders for single-cell gene expression data. Bioinformatics. 2020. https://doi.org/10.1093/bioinformatics/btaa293.
21. Tran D, Nguyen H, Tran B, La Vecchia C, Luu HN, Nguyen T. Fast and precise single-cell data analysis using a hierarchical autoencoder. Nat Commun. 2021;12(1):1–10. https://doi.org/10.1038/s41467-021-21312-2.
22. Rousseeuw J. A graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1989;20:53–65.
23. Bica I, Andrés-Terré H, Cvejic A, Liò P. Unsupervised generative and graph representation learning for modelling cell differentiation. Sci Rep. 2020;10(1):1–13. https://doi.org/10.1038/s41598-020-66166-8.
24. Wang D, Gu J. VASC: dimension reduction and visualization of single-cell RNA-seq data by deep variational autoencoder. Genom Proteom Bioinf. 2018;16(5):320–31. https://doi.org/10.1016/j.gpb.2018.08.003.
25. Lin E, Mukherjee S, Kannan S. A deep adversarial variational autoencoder model for dimensionality reduction in single-cell RNA sequencing analysis. BMC Bioinformatics. 2020;21(1):1–11. https://doi.org/10.1186/s12859-020-3401-5.
26. Geddes TA, Kim T, Nan L, Burchfield JG, Yang JY, Tao D, Yang P. Autoencoder-based cluster ensembles for single-cell RNA-seq data analysis. BMC Bioinformatics. 2019;20(19):660. https://doi.org/10.1186/s12859-019-3179-5.
27. Talwar D, Mongia A, Sengupta D, Majumdar A. AutoImpute: autoencoder based imputation of single-cell RNA-seq data. Sci Rep. 2018;8(1):16329. https://doi.org/10.1038/s41598-018-34688-x.
28. Sun S, Liu Y, Shang X. Deep generative autoencoder for low-dimensional embedding extraction from single-cell RNAseq data. In: Proceedings of the IEEE international conference on bioinformatics and biomedicine. IEEE. 2019; pp. 1365–1372. https://doi.org/10.1109/BIBM47256.2019.8983289.

29. Badsha MB, Li R, Liu B, Li YI, Xian M, Banovich NE, Fu AQ. Imputation of single-cell gene expression with an autoencoder neural network. Quant Biol. 2020. https://doi.org/10.1007/s40484-019-0192-7.
30. Rao J, Zhou X, Lu Y, Zhao H, Yang Y. Imputing single-cell RNA-seq data by combining graph convolution and autoencoder neural networks. BioRxiv. 2020. https://doi.org/10.1101/2020.02.05.935296.
31. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. 2018;19(1):15. https://doi.org/10.1186/s13059-017-1382-0.
32. Satija R, Farrell JA, Gennert D, Schier AF, Regev A. Spatial reconstruction of single-cell gene expression data. Nat Biotechnol. 2015;33(5):495. https://doi.org/10.1038/nbt.3192.
33. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, et al. Tensorflow: a system for large-scale machine learning. In: Proceedings of the symposium on operating systems design and implementation; 2016. p. 265–283.
34. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A. Automatic differentiation in PyTorch. In: Proceedings of the conference on advances in neural information processing systems; 2017.
35. Zhao S, Song J, Ermon S. Infovae: balancing learning and inference in variational autoencoders. Proc AAAI Conf Artif Intell. 2019;33:5885–92.
36. Polański K, Young MD, Miao Z, Meyer KB, Teichmann SA, Park J-E. BBKNN: fast batch alignment of single cell transcriptomes. Bioinformatics. 2020;36(3):964–5. https://doi.org/10.1093/bioinformatics/btz625.
37. Korsunsky I, Millard N, 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. Nat Methods. 2019;16:1289–96. https://doi.org/10.1038/s41592-019-0619-0.
38. Johnson WE, Li C, Rabinovic A. Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics. 2007;8(1):118–27. https://doi.org/10.1093/biostatistics/kxj037.
39. Leek JT, Johnson WE, Parker HS, Jaffe AE, Storey JD. The `sva` package for removing batch effects and other unwanted variation in high-throughput experiments. Bioinformatics. 2012;28(6):882–3. https://doi.org/10.1093/bioinformatics/bts034.
40. Pedersen B. Python implementation of ComBat. GitHub; 2012.
41. Kang HM, Subramaniam M, Targ S, Nguyen M, Maliskova L, McCarthy E, Wan E, Wong S, Byrnes L, Lanata CM, et al. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. Nat Biotechnol. 2018;36(1):89. https://doi.org/10.1038/nbt.4042.
42. Grün D, Muraro MJ, Boisset J-C, Wiebrands K, Lyubimova A, Dharmadhikari G, van den Born M, van Es J, Jansen E, Clevers H, et al. De novo prediction of stem cell identity using single-cell transcriptome data. Cell Stem Cell. 2016;19(2):266–77. https://doi.org/10.1016/j.stem.2016.05.010.
43. Muraro MJ, Dharmadhikari G, Grün D, Groen N, Dielen T, Jansen E, van Gurp L, Engelse MA, Carlotti F, de Koning EJ, et al. A single-cell transcriptome atlas of the human pancreas. Cell Syst. 2016;3(4):385–94. https://doi.org/10.1016/j.cels.2016.09.002.
44. Lawlor N, George J, Bolisetty M, Kursawe R, Sun L, Sivakamasundari V, Kycia I, Robson P, Stitzel ML. Single-cell transcriptomes identify human islet cell signatures and reveal cell-type-specific expression changes in type 2 diabetes. Genome Res. 2017;27(2):208–22. https://doi.org/10.1101/gr.212720.116.
45. Segerstolpe Å, Palasantza A, Eliasson P, Andersson E-M, Andréasson A-C, Sun X, Picelli S, Sabirsh A, Clausen M, Bjursell MK, et al. Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes. Cell Metab. 2016;24(4):593–607. https://doi.org/10.1016/j.cmet.2016.08.020.
46. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM III, Hao Y, Stoeckius M, Smibert P, Satija R. Comprehensive integration of single-cell data. Cell. 2019. https://doi.org/10.1016/j.cell.2019.05.031.
47. Han X, Wang R, Zhou Y, Fei L, Sun H, Lai S, Saadatpour A, Zhou Z, Chen H, Ye F, et al. Mapping the mouse cell atlas by microwell-seq. Cell. 2018;172(5):1091–107. https://doi.org/10.1016/j.cell.2018.02.001.
48. Consortium TM. et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature. 2018;562:367–372. https://doi.org/10.1038/s41586-018-0590-4
49. Hubert L, Arabie P. Comparing partitions. J Classif. 1985;2(1):193–218. https://doi.org/10.1007/BF01908075.
50. Strehl A, Ghosh J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J Mach Learn Res. 2002;3:583–617.
51. Fowlkes EB, Mallows CL. A method for comparing two hierarchical clusterings. J Am Stat Assoc. 1983;78(383):553–69.
52. Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res. 2010;11:2837–54.
53. Rosenberg A, Hirschberg J. V-measure: a conditional entropy-based external cluster evaluation measure. In: Proceedings of the conference on empirical methods in natural language processing and computational natural language learning; 2007. p. 410–420.
54. Macosko EZ, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, Tirosh I, Bialas AR, Kamitaki N, Martersteck EM, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell. 2015;161(5):1202–14. https://doi.org/10.1016/j.cell.2015.05.002.
55. Klein AM, Mazutis L, Akartuna I, Tallapragada N, Veres A, Li V, Peshkin L, Weitz DA, Kirschner MW. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell. 2015;161(5):1187–201. https://doi.org/10.1016/j.cell.2015.04.044.
56. Hashimshony T, Wagner F, Sher N, Yanai I. CEL-Seq: single-cell RNA-Seq by multiplexed linear amplification. Cell Rep. 2012;2(3):666–73. https://doi.org/10.1016/j.celrep.2012.08.003.
57. Hashimshony T, Senderovich N, Avital G, Klochendler A, de Leeuw Y, Anavy L, Gennert D, Li S, Livak KJ, Rozenblatt-Rosen O, et al. CEL-Seq2: sensitive highly-multiplexed single-cell RNA-Seq. Genome Biol. 2016;17(1):77. https://doi.org/10.1186/s13059-016-0938-8.
58. Zheng GX, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, Ziraldo SB, Wheeler TD, McDermott GP, Zhu J, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun. 2017;8:14049. https://doi.org/10.1038/ncomms14049.

59. Gierahn TM, Wadsworth MH II, Hughes TK, Bryson BD, Butler A, Satija R, Fortune S, Love JC, Shalek AK. Seq-Well: portable, low-cost RNA sequencing of single cells at high throughput. Nat Methods. 2017;14(4):395. https://doi.org/10.1038/nmeth.4179.
60. Islam S, Kjällquist U, Moliner A, Zajac P, Fan J-B, Lönnerberg P, Linnarsson S. Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq. Genome Res. 2011;21(7):1160–7. https://doi.org/10.1101/gr.110882.110.
61. Ramsköld D, Luo S, Wang Y-C, Li R, Deng Q, Faridani OR, Daniels GA, Khrebtukova I, Loring JF, Laurent LC, et al. Full-length mRNA-Seq from single-cell levels of RNA and individual circulating tumor cells. Nat Biotechnol. 2012;30(8):777. https://doi.org/10.1038/nbt.2282.
62. Picelli S, Faridani OR, Björklund ÅK, Winberg G, Sagasser S, Sandberg R. Full-length RNA-seq from single cells using smart-seq2. Nat Protoc. 2014;9(1):171. https://doi.org/10.1038/nprot.2014.006.
63. Jaitin DA, Kenigsberg E, Keren-Shaul H, Elefant N, Paul F, Zaretsky I, Mildner A, Cohen N, Jung S, Tanay A, et al. Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types. Science. 2014;343(6172):776–9. https://doi.org/10.1126/science.1247651.
64. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep. 2019. https://doi.org/10.1038/s41598-019-41695-z.
65. Mann HB, Whitney DR. On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat. 1947;50–60.
66. Wilcoxon F. Individual comparisons by ranking methods. In: Breakthroughs in statistics. New York: Springer;1992. p. 196–202. https://doi.org/10.1007/978-1-4612-4380-9_16.
67. Dunn OJ. Multiple comparisons among means. J Am Stat Assoc. 1961;56(293):52–64.
68. Ma F, Pellegrini M. ACTINN: automated identification of cell types in single cell RNA sequencing. Bioinformatics. 2019. https://doi.org/10.1093/bioinformatics/btz592.
69. Ranzoni AM, Tangherloni A, Berest I, Riva SG, Myers B, Strzelecka PM, Xu J, Panada E, Mohorianu I, Zaugg JB, et al. Integrative single-cell rna-seq and atac-seq analysis of human developmental hematopoiesis. Cell Stem Cell. 2021;28(3):472–87. https://doi.org/10.1016/j.stem.2020.11.015.
70. Simidjievski N, Bodnar C, Tariq I, Scherer P, Andres Terre H, Shams Z, Jamnik M, Liò P. Variational autoencoders for cancer data integration: design principles and computational practice. Front Genet. 2019;10:1205. https://doi.org/10.3389/fgene.2019.01205.
71. Trębacz M, Shams Z, Jamnik M, Scherer P, Simidjievski N, Terre HA, Liò P. Using ontology embeddings for structural inductive bias in gene expression data analysis; 2020. arXiv preprint arXiv:2011.10998.
72. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, et al. Gene ontology: tool for the unification of biology. Nat Genet. 2000;25(1):25–9.
73. Raudvere U, Kolberg L, Kuzmin I, Arak T, Adler P, Peterson H, Vilo J. g:Profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update). Nucl Acids Res. 2019;47(W1):191–8. https://doi.org/10.1093/nar/gkz369.
74. Chen X, Miragaia RJ, Natarajan KN, Teichmann SA. A rapid and robust method for single cell chromatin accessibility profiling. Nat Commun. 2018;9(1):5345. https://doi.org/10.1038/s41467-018-07771-0.
75. Gretton A, Borgwardt K, Rasch M, Schölkopf B, Smola AJ. A kernel method for the two-sample-problem. In: Proceedings of the conference on advances in neural information processing systems;2007. p. 513–520.
76. Kullback S, Leibler RA. On information and sufficiency. Ann Math Stat. 1951;22(1):79–86. https://doi.org/10.1214/aoms/1177729694.
77. Risso D, Perraudeau F, Gribkova S, Dudoit S, Vert J-P. A general and flexible method for signal extraction from single-cell RNA-seq data. Nat Commun. 2018;9(1):1–17. https://doi.org/10.1038/s41467-017-02554-5.
78. Hie B, Bryson B, Berger B. Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. Nat Biotechnol. 2019;37(6):685–91. https://doi.org/10.1038/s41587-019-0113-3.
79. Haghverdi L, Lun AT, Morgan MD, Marioni JC. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nat Biotechnol. 2018;36(5):421. https://doi.org/10.1038/nbt.4091.
80. Kingma DP, Ba J. Adam: A method for stochastic optimization; 2014. arXiv preprint arXiv:1412.6980.

Publisher's Note