An Ensemble Machine Learning Approach for Benchmarking and Selection of scRNA-seq Integration Methods

Konghao Zhao, Sapan Bhandari, Nathan P. Whitener, Jason M. Grayson and Natalia Khuri
Wake Forest University
Winston Salem, North Carolina, USA
natalia.khuri@wfu.edu

ABSTRACT

Accurate integration of high-dimensional single-cell sequencing datasets is important for the construction of cell atlases and for the discovery of biomarkers. Because the performance of integration methods varies in different scenarios and on different datasets, it is important to provide end users with an automated system for the benchmarking and selection of the best integration among several alternatives. Here, we present a system that uses an ensemble of auditors, trained by supervised machine learning, which quantifies residual variability of integrated data and automatically selects the integration with the smallest difference between observed and expected batch effects. A rigorous and systematic validation was performed using 6 popular integration methods and 52 benchmark datasets. Algorithmic and data biases were uncovered and shortcomings of existing validation metrics were examined. Our results demonstrate the utility, validity, flexibility and consistency of the proposed approach.

CCS CONCEPTS

• **Applied computing** → **Bioinformatics**; • **Computing methodologies** → **Supervised learning by classification**.

KEYWORDS

Benchmarking, classification, data integration, machine learning, single-cell RNA sequencing

ACM Reference Format:
Konghao Zhao, Sapan Bhandari, Nathan P. Whitener, Jason M. Grayson and Natalia Khuri. 2023. An Ensemble Machine Learning Approach for Benchmarking and Selection of scRNA-seq Integration Methods. In *14th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '23), September 3–6, 2023, Houston, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3584371.3613072

1 INTRODUCTION

Due to the improvements in efficiency, scalability and affordability of single-cell RNA-sequencing (scRNA-seq) technologies, it is now possible to study the complexity and heterogeneity of gene expression within individual cells, as well as the composition of cell types and cell states within different tissues, organs and organisms [15, 21, 30, 38]. Often, multiple scRNA-seq datasets are integrated together to increase the statistical power of data mining [4, 29] and to create cell atlases, such as the Human Cell Atlas [25], for example. The integrated datasets may come from different biological samples, donors, sequencing centers or sequencing platforms. Collectively, the complex technical variations in sequencing protocols, collection time, sample acquisition and handling, reagents and data preprocessing are known as the batch effects [14, 19, 34].

Computationally, integration of scRNA-seq datasets involves two optimization objectives. Firstly, biologically similar cells from different datasets should be well mixed to remove technical variations due to batch effects. Secondly, biologically dissimilar cells should be well separated in the integrated data to preserve their distinctness. Optimizing for these two objectives is challenging, in part because true technical and biological variability of integrated datasets is not always known.

Past benchmarking studies used a combination of qualitative and quantitative approaches to evaluate existing integration methods [8, 19, 33]. Although past benchmarking studies provided useful information about the strengths and weaknesses of different integration methods, such as their performance, computational run-time, and memory requirements, they also revealed that no integration method outperforms other methods in all benchmarking scenarios and on all benchmark datasets [8, 19, 33]. It was also found that the performance of different methods was greatly influenced by the choice of pre-integration and post-integration techniques for highly variable gene (HVG) selection, normalization, scaling, dimensionality reduction and cluster analysis [19]. Given that methods perform differently for different integration scenarios, datasets and data preparation approaches, and that past benchmarking results present only a static view of existing integration methods, it is challenging for the end users to decide on the best integration for their specific data mining needs.

Here, we present a new approach, called Machine Learning Based Integration Method Selection (ML-IMS), for the evaluation of integration results and benchmarking of existing and emergent integration tools. ML-IMS trains integration auditors, which estimate residual variability in low-dimensional matrices of integrated data. The primary objective of ML-IMS is to evaluate the degree to which different datasets (batches) have been mixed together. Specifically, the auditors are trained to predict identities of the batches using a balanced subset of integrated data, and trained models are used to label the low-dimensional representation of cells in the withheld subset of integrated data (Figure 1). The main idea behind this is that if cells are well-mixed, it should be difficult to predict which batch

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
BCB '23, September 3–6, 2023, Houston, TX, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0126-9/23/09…$15.00
https://doi.org/10.1145/3584371.3613072

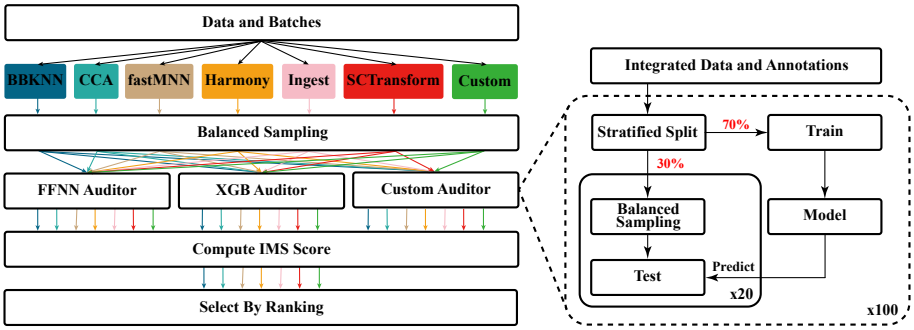

Figure 1: Architecture of Machine Learning Based Integration Method Selection. Shown are the main steps of the proposed system for benchmarking and automated selection of the best integration. ML-IMS inputs low-dimensional integrated data, executes ML audits and computes IMS scores for each input. By default, the integration with the smallest IMS score is selected.

they came from, beyond a random chance. If, however, the integration fails to correctly mix biologically similar cells from different batches, identities of individual batches of cells can be predicted more accurately than by random chance. When presented with several low-dimensional integrated datasets, ML-IMS performs the audit and automatically selects the one with the smallest difference between its observed and expected classification accuracy. Notably, the same approach can be used to examine how well the biological variability is preserved. Instead of using batch labels, ML-IMS can use biological cell types, for example, to train the auditors and select integrated data with the greatest difference between expected and observed classification accuracy.

In this work, we made the following contributions. First, we designed and implemented an automated system for the evaluation of residual variability of integrated data and selection of the best integration result. The system is implemented as an open-source, extendable package in the Python programming language, which accepts as input low-dimensional integrated data and their class labels and outputs IMS scores. The IMS score measures the difference between the expected classification accuracy and the observed classification accuracy, averaged across class labels and across an ensemble of auditors. Second, we validated ML-IMS in rigorous and systematic experiments using 6 popular integration methods and 52 datasets, and compared and contrasted IMS with existing metrics, such as kBET [5] and Lisi [17, 18]. Our experimental results can be used in future benchmarking studies of emergent integration methods. Third, we demonstrated that ML-IMS can detect algorithmic biases of integration methods as well as data biases caused by different sequencing technologies, preprocessing and experimental conditions at individual sequencing centers.

2 PRIOR AND RELEVANT WORK

Approaches for the evaluation of integration results can be divided into two groups. The first group focuses on the evaluation of methods and the second group focuses on the evaluation of results by quantifying the quality of the integrated data.

Several benchmarking studies of integration methods have been recently performed, varying in their experimental designs, datasets, evaluation criteria and metrics. In one of the earliest benchmarking studies [33], 10 datasets were integrated using 5 different scenarios, such as the integration of non-identical cell types, large datasets and simulated datasets. Fourteen integration methods were reviewed for their performance, scalability and computational time. Later, as part of an effort to create a reference scRNA-seq dataset, 2 well-characterized, commercially-available cell lines were sequenced on 4 different platforms, at 4 different sequencing centers and preprocessed in 7 different ways. Performance of 7 different integration methods was then examined using these datasets and their mixtures [8, 9]. More recently, a comprehensive benchmarking study of integration methods for the construction of cell atlases was performed that evaluated 68 combinations of methods and preprocessing techniques on 85 batches [19].

The main outcome of past benchmarking studies is the recognition that different integration tasks, different datasets, and different preprocessing methods produce different results, and that no integration method performs well in all scenarios. Some of the challenges noted in prior studies include differences in integration's inputs and outputs, inconsistencies in preprocessing, lack of good benchmarking datasets and evaluation metrics, and poor accessibility and adaptability to the needs of end users. The majority of these limitations continue to persist, although some recent works addressed some of them. For example, to support the evaluation of new integration methods and to improve adaptability to new benchmark datasets, BatchBench package provides programmatic support for the creation of a pipeline for different benchmarking experiments [6]. Similarly, scrnabench package comprises a suite of benchmarking datasets and standardized benchmarking workflows, which include data preparation and integration [35]. While both of these packages make benchmarking more accessible and adaptable, they are still limited by the lack of objective and interpretable metrics for the evaluation of integration results.

There are two computational approaches for the assessment of integration results, qualitative evaluation and quantitative validation. Qualitative evaluation by visualization is commonly used [5, 6, 8, 33, 39]. However, the interpretation of visualizations is subjective and only possible for a small number of cells. Because integrated

data are projected onto a low-dimensional space, visualizations may not capture all aspects of the data, potentially ignoring subtle variations or patterns that could impact the assessments. Additionally, datasets integrated using different methods may exhibit different shapes in the low-dimensional space, making it challenging for the interpreter to distinguish the minor differences between the results.

Entropy is an example of a quantitative metric. It was used in early benchmarking studies [2, 6, 39]. It is calculated by averaging the local Entropy of each batch, where local Entropy is calculated as the negative sum of the probability that a cell has a certain batch in its local neighborhood multiplied by the logarithm of that probability [27]. A high entropy means a homogeneous mixture of the batches. In later studies, two new quantitative metrics were proposed, namely kBET [5] and Lisi [18]. Past benchmarking studies, as well as the results of this work, noted that kBET and Lisi can give contradictory assessments, making it challenging to decide on the best integration approach.

Ranging from 0 to 1, kBET estimates residual batch effects by performing repeated Pearson's χ^2 test based on the randomly selected neighborhoods, and the results are averaged as a rejection rate over all tested neighborhoods. In other words, kBET determines whether the batch composition of a neighborhood of a cell is similar to the expected composition. To do so, it constructs a neighborhood graph, where the size of the neighborhood is a user-defined parameter.

Lisi estimates batch effects by computing a diversity score, which ranges from 0 to the number of batches. Specifically, Lisi uses Gaussian kernel-based distribution to construct neighborhoods with distance-based weights for cells [18]. It measures the effective number of batches in a local neighborhood and computes a diversity score for each cell. It has been recently shown that even with the well-mixed local neighborhoods, the value of Lisi may be significantly smaller than the number of batches, which is the optimal value for a well-mixed dataset, due to the imbalance in sizes of integrated datasets [18].

The main shortcoming of Entropy, kBET and Lisi is their reliance on distance computation, which is needed to define local neighborhoods and find cells within those neighborhoods. Limitations of distance-based approaches are well-known and include, for example, the choice of the best distance metric, the curse of dimensionality, arbitrary selection of the neighborhood radii, and so on. Specific to the integration tasks, kBET and Lisi may overestimate the performance of methods that use graph-based approaches in the integration of the datasets and underestimate the performance of methods that do not rely on the construction of the neighborhood graphs.

Our proposed method can be used for benchmarking of new and existing integration methods as well as for the evaluation of the quality of integrated datasets. Unlike kBET and Lisi, ML-IMS does not rely on the computation of neighborhood graphs, thus providing an orthogonal information about the performance of distance-based integration methods.

3 METHODS AND DATA

3.1 Proposed Method

ML-IMS is an automated and extendable approach for the selection of the best integration. Given several datasets, which are integrated using different methods, along with their labels, ML-IMS performs supervised audits to quantify the residual variability of each class label. An aggregated audit score is computed by averaging the classification results of an ensemble of auditors and across class labels (Figure 1).

The audit is conducted as follows. First, each integrated dataset is balanced by random sampling of an equal number of cells of each class label, without replacement. During the audit, integrated data are repeatedly divided into balanced training (70%) and test (30%) subsets, and the training data are used to train several ML classifiers. Following a previously established protocol [32], we repeatedly sample 20% of cells from the withheld test subset and predict their labels using each of the trained classifiers. Sampling from test data is repeated 20 times, after which integrated data are divided again into training and test subsets, and the audit is repeated. The number of repeated iterations is determined by the user. In this work, we repeat each audit 100 times to obtain robust estimates. Because training and test sets are balanced, intuitive metric of classification accuracy can be computed for each class label, measuring the number of correctly predicted labels of each class divided by the total number of cells of each class. Classification accuracy is averaged over all test subsets and over the total number of iterations.

To facilitate an automated selection of the integrated dataset, an Integration Method Selection (IMS) score is computed as follows. Given n batches and m auditors, the difference between the observed classification accuracy of batch i by auditor j, obs_{ij}, and the baseline expected accuracy, $\frac{1}{n}$, is calculated. These differences are averaged for n batches and m auditors (Equation 1).

$$\text{IMS} = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{n}\sum_{i=1}^{n}\left(obs_{ij} - \frac{1}{n}\right)\right] \tag{1}$$

When class labels are batches, integrated dataset with the lowest score is selected. When, on the other hand, class labels are biological cell types, for example, integrated dataset with the highest score is automatically selected.

3.2 Implementation Details

ML-IMS is an open source and publicly available at https://github.com/SheltonZhaoK/ML-IMS. The modular architecture of the package allows for an easy substitution, removal or addition of auditors. By default, ML-IMS includes two auditors, namely, a feed forward neural network (FFNN) and an eXtreme gradient boosting (XGB) tree. These algorithms are selected due to their accuracy of cell-type prediction [3, 20, 24, 31] and their parameters were tuned as previously described [3].

FFNN auditors are implemented as the multi-class predictors using the Tensorflow and Keras libraries [1]. FFNN comprises input layer, one hidden layer with 512 hidden nodes and output layer. The number of nodes in the input layer is set to the number of data dimensions, and the number of output nodes is equal to the number of classes. ReLU activation function is used in the hidden layer and softmax activation function is used in the output layer. Categorical cross-entropy is used as the loss function because the network has more than 2 class labels. In addition, RMSprop optimizer is used

with the learning rate of 1E-4. The models are trained in batches of size 128 for 50 epochs.

XGB auditors are implemented as the multi-class classifiers using XGBoost package [7]. The following parameters are used to train the classifiers: 'max_depth': 4, 'eta': 1, 'objective': 'multi:softprob', 'eval_metric': 'mlogloss', and 'num_round': 50. Gradient boosting is the process of combining weak learning decision trees to create a stronger model.

To demonstrate how ML-IMS can be extended to include additional auditors, we also implement a distance-based k-Nearest Neighbor (KNN) classifier using KNeighborsClassifier function from the Scikit-learn package [22]. We use Euclidean distances and set k to 15 [3].

ML-IMS implementation also includes support for parallel execution of audits, which is implemented using Python's multiprocessing module. In this work, all experiments are performed on a High Performance Computing cluster.

3.3 Datasets

Three data sources are used in this study. We use 48 benchmark datasets of 2 cell lines from the scrnabench package [35], 3 datasets from the SeuratData package [26] and 1 dataset from a recent benchmarking study [19].

The 48 benchmark datasets are extracted from scrnabench, which links to their original repository [9]. These data are from 2 cell lines derived from breast cancer (cell line A) and normal B lymphocytes (cell line B). The number of sequenced cells ranges from 59 to 6,097, and the number of genes ranges from 16,931 to 32,502. Each cell is annotated with its cell line, sequencing technology (10X, C1, C1_HT, ICELL8), center (LLU, NCI, FDA, TBU), and preprocessing method (cellranger2.0, cellranger3.1, zumi, umitools, featureCounts, rsem, kallisto). In addition, each cell is labeled with the name of its dataset and a unique cell identification number.

We also retrieve 3 benchmark datasets from the SeuratData package that stores sequenced data and their annotations. Specifically, we use panc8 [11] benchmark of 8 datasets of human pancreas cells sequenced on 5 different platforms. The entire collection contains 14,890 cells and 14,890 genes. Due to the limited amount of cells, cells sequenced by fluidigmc1 technology are excluded, leaving 4 datasets (celseq, celseq2, smartseq2, indrop). The second benchmark dataset, pbmcsca [10], consists of 31,021 peripheral blood mononuclear cells (PBMC) and 14,053 genes, sequenced on 9 different platforms. After removing batches with low counts of cells, 7 batches (10x Chromium (v2), 10x Chromium (v2) A, 10x Chromium (v2) B, 10x Chromium (v3), Drop-seq, Seq-Well, inDrops) are retained. Finally, we use the ifnb benchmark [16] comprising 14,000 PBMCs with 14,053 genes. Cells are either stimulated by IFN-β or left untreated, resulting in two class labels (STIM, CTRL). The last dataset, bone_marrow is the cell-atlas benchmark used in the past benchmarking study [19]. It comprises 7 samples (SIGAG8, SIGAF1, SIGAH1, SIGAD1, BoneMarrowcKit1_dge, BoneMarrowcKit2_dge, BoneMarrowcKit3_dge) of mouse cells, sequenced using 2 different technologies (10X Genomics and Microwell-seq).

All datasets are preprocessed and integrated using the same, previously published workflows [37] implemented in scrnabench [35] and Scanpy [36] packages. We compare 6 integration methods: 4 methods are implemented in the R programming language, namely CCA [28], fastMNN [13], Harmony [17], and SCTransform [12], and 2 methods are implemented in the Python programming language, namely BBKNN [23] and Ingest [36]. For integration performed in Scanpy, benchmarks are first prepared using scrnabench's preparation workflow, then converted to .h5ad format, imported into Scanpy and integrated. This ensures that all data are prepared using the same standard workflow.

The scrnabench workflow loads each dataset into a Seurat object and filters them as follows. Cells containing fewer than 200 genes are removed and genes detected in fewer than 2 cells are filtered out. Next, the total number of mRNAs and total number of genes are counted for each cell, and cells with the total number of mRNAs and total number of genes outside of 2 standard deviations from the mean counts are removed. Additionally, cells with more than 10% mitochondrial counts are filtered out and top 2,000 HVGs are selected.

For BBKNN workflow, matrix of HVGs is converted to .h5ad format and read into Scanpy. Then, sc.pp.scale function is called to scale the dataset, which is further reduced to 10 principal components by the sc.tl.pca function. The integration is performed using the sc.external.pp.bbknn function and reduced to 2 UMAP embeddings. The CCA integration works with the count matrix of HVGs, by finding integration anchors and then integrating the data. The k.filter parameter is changed from the default 200 to the number of cells in the smallest dataset to allow for the integration of smaller datasets. Then, integrated data are scaled and subjected to a dimensionality reduction using PCA, followed by a non-linear reduction using UMAP. FastMNN is an integration that combines scaling and normalization. Therefore, the matrix of HGVs is directly integrated with batch names as input, followed by the UMAP reduction. For Harmony integration workflow, the count matrix of HVGs is scaled and reduced to 10 principal components. Harmony integration is performed using datasets' names as the batch identifiers, and harmonized embeddings are reduced to 2 UMAP components. The Ingest integration workflow integrates HVGs and annotations of batches using the first batch as a reference, by default. It reduces the first batch by PCA, constructs a kNN graph, and reduces it to UMAP embeddings. Then, it processes other batches similarly, repeatedly mapping batches to the reference batch by the sc.tl.ingest function, and finally concatenates the data. SCTransform integration workflow follows a process similar to CCA, except that it executes Seurat's SelectIntegrationFeatures function, followed by PrepSCTIntegration. Integrated data are reduced to 2 UMAP embeddings.

Thus, the outputs of all integration methods are 2-dimensional UMAP embeddings and cells' annotations, which are saved as separate comma-separated values (csv) files. Cell's unique identifier is used as a key to cross-reference the UMAP and the annotation data. These 2 csv files are used as inputs to the ML-IMS package.

3.4 Additional Metrics

For comparison, we compute Lisi and kBET. Lisi scores range from 0 to the number of batches. To compare these scores to IMS, we normalize them by dividing Lisi by the total number of batches, $\frac{lisi}{n}$. Therefore, normalized scores range from 0 to 1, where higher

score represents a better quality of integration. kBET computes the overall averaged rejection rate across 100 tests. Therefore, in order to ensure a fair comparison with ML-IMS, we average kBET P-values of cells across all batches. Thus, kBET ranges from 0 to 1, where higher score represents a higher confidence that integrated data are well-mixed.

4 RESULTS

Our results demonstrate that ML-IMS can detect algorithmic biases in the integration methods. Here, the algorithmic bias refers to systematic and repeatable errors in an integration method that consistently "favors" one of the batches over the others. Consider the following proof-of-concept experiment, in which we integrate 4 identical copies of each of the 48 benchmarks from the scrnabench package. After the integration, we reduce each dataset to 2 UMAP embeddings. If the integration process is unbiased, dataset's copies should be well-mixed, and it should be challenging to discern, beyond a random guess of 0.25, the identity of these 4 copies. A biased integration, on the other hand, will manifest itself when the auditors, trained with UMAP data, classify cells of one (or more) of the copies with an accuracy greater than 0.25.

We run this experiment using two ML auditors, FFNN and XGB, and perform 100 audits of 48 integrated datasets, each with 20 randomly sampled and balanced tests. FFNN and XGB are selected because they are used in previously reported cell-type predictors, and because their algorithms are not distance-based. Our benchmark datasets vary in complexity, size, sequencing platform and preprocessing type, allowing us to isolate algorithmic biases from batch effects. Each benchmark comprises cells from either normal or cancer cell line. To obtain robust estimates, for each duplicated copy of a dataset, we average classification accuracy scores of 2,000 repeated training and test experiments.

Our results show that all integration methods, except Ingest, successfully mix duplicated copies and these copies are indistinguishable in the integrated data, beyond the random chance. The average accuracy of classifying copies across all 48 datasets is below 0.25, regardless of the sequencing method, center, preprocessing or size. The average accuracy of classification using FFNN is 0.21 ± 0.02, 0.20 ± 0.02, 0.22 ± 0.02, 0.21 ± 0.03, per copy, and the average accuracy of XGB is 0.14 ± 0.04, 0.14 ± 0.05, 0.14 ± 0.05, 0.14 ± 0.04 (Figure 2).

On the other hand, audits of Ingest integrations show that the accuracy of classification is notably and consistently higher than 0.25 for the first copy and smaller than the random guess for the remaining 3 copies (Figure 2E). Specifically, the accuracy of classification of the first copy is 0.31 (FFNN) and 0.46 (XGB), whereas the classification accuracy of each of the other copies is 0.21 (FFNN) and 0.04 (XGB). These results point to an algorithmic bias in the decision-making process of Ingest method. This bias may be due to Ingest's approach of using the first copy as a reference batch [36]. Moreover, these results are reproducible. Even when the copy labels are swapped prior to the audits, ML-IMS consistently classifies the same copy more accurately than expected by chance. Notably, although CCA and SCTransform also use the first copy as the reference, they do not introduce algorithmic biases in their integration results.

We note that the average per-copy classification accuracy of FFNN audits is higher than of XGB, and the standard deviation is smaller, for all integration methods except Ingest. Also, the average per-copy accuracy of FFNN is slightly higher than the baseline for some cancer benchmarks, such as the benchmarks sequenced using 10X Genomics at NCI and preprocessed using cellranger3.1 and zumi. These differences, however, are not statistically significant. On the other hand, for Ingest, XGB auditor has significantly greater classification accuracy of the first copy compared with FFNN-based auditor, reaching above 0.89 in some of the benchmarks (Figure 2E). These benchmarks are all from datasets sequenced using C1 technology at LLU. In addition, Ingest's integration of 10X Genomics datasets has smaller algorithmic bias compared with datasets sequenced using C1, C1_HT and ICELL8.

To validate these results, we compute kBET, and Lisi metrics. In well-mixed datasets with 4 batches, kBET P-value and normalized Lisi should be close to 1. Indeed, the average kBET P-value and normalized Lisi on 48 benchmark datasets for all integration methods, except Ingest, are 1.00 ± 0.00 and 0.96 ± 0.01. For Ingest, kBET P-value and normalized Lisi are 0.96 ± 0.09, and 0.90 ± 0.04. While there are drops in the average kBET and Lisi scores, it is difficult to interpret them because no threshold exists for these scores which can separate good integration quality from poor. ML-IMS, on the other hand, provides an easily interpretable and comparable classification score as well as the threshold of what is expected by random chance in a balanced dataset.

While per-batch classification accuracy helps detect algorithmic biases, it may not always be practical or useful to focus on the individual batches. Therefore, we aggregate per-batch accuracy values by averaging them into a single classification accuracy score, which can be used for the comparison and ranking of different integration results. Thus, average classification accuracy higher than expected by random chance implies larger residual batch effects in the integrated data and lower average accuracy means better integration results. The trade-off is that averaged accuracy may miss algorithmic biases, such as the one discovered in Ingest's integration.

To illustrate how to use the averaged classification accuracy to detect batch effects due to data preprocessing, we integrate 48 scrnabench datasets as follows. We construct 14 groups of datasets, where each group comprises datasets of the same cell line (cancer or normal), sequenced in the same center (LLU, NCI, FDA or TBU) and on the same platform (10X Genomics, C1, C1_HT, or ICELL8) but preprocessed differently. Four different preprocessing methods were used for 10X Genomics data and three different methods were used for other sequencing platforms. We integrate these 14 groups using preprocessing as the class label. Therefore, the expected classification accuracy for 10X integrations is 0.25 and for all other integrations, it is 0.33.

Our results reveal several patterns. First, integration of cells of cancer line is more challenging than the integration of normal cells (Figure 3). Our auditors detect more uncorrected batch effects in the cancer cell line. Overall, the classification scores of the integrated cancer cells are 0.30 (FFNN) and 0.27 (XGB) compared to 0.28 (FFNN) and 0.25 (XGB) for the normal cells. This means that different preprocessing methods produce significantly different gene expression profiles of the same scRNA-seq data. These variations

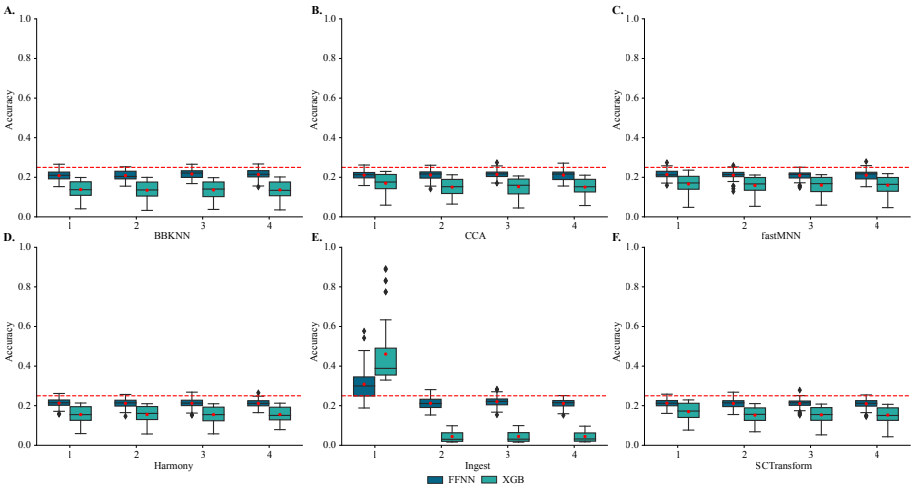

Figure 2: Audit of algorithmic biases of 6 integration methods on 48 benchmark datasets. Shown are the box plots of the distributions of classification accuracy scores of each of the 4 copies in the data integrated by (A) BBKNN, (B) CCA, (C) fastMNN, (D) Harmony, (E) Ingest, and (D) SCTransform. Red dashed line shows the expected classification accuracy of 0.25. Audits are performed 100 times for each dataset and results are averaged.

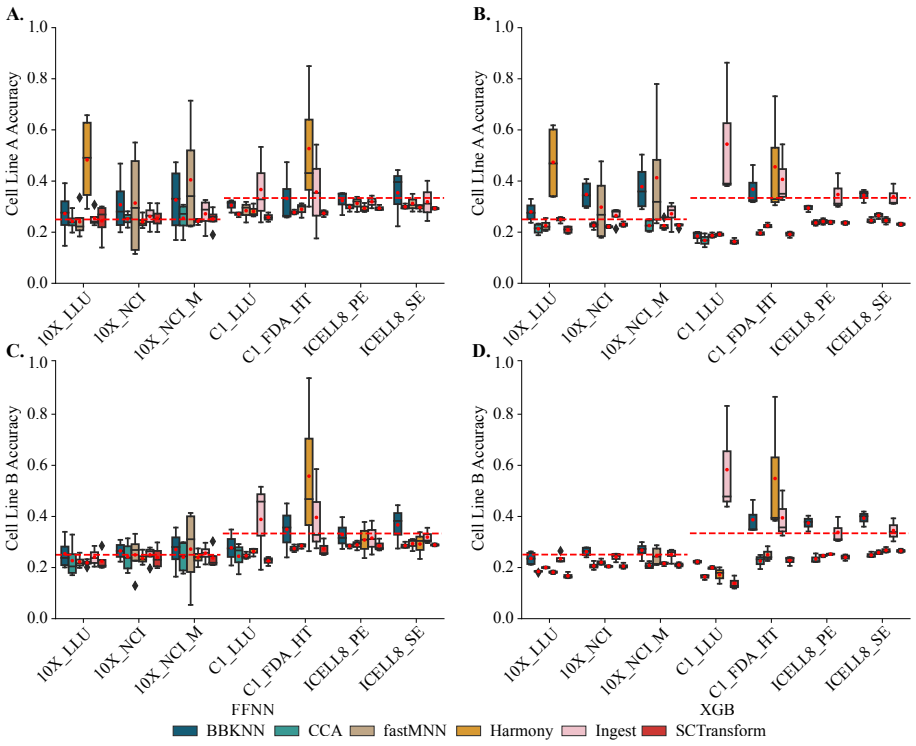

Figure 3: Preprocessing audit of 6 integration methods on 48 benchmark datasets. Shown are the box plots of the distributions of classification accuracy scores averaged across all class labels of each integrated dataset from same centers and platforms but different preprocessing techniques. Datasets from 10X have 4 preprocessing methods and expected value of $\frac{1}{4}$ (red dashed line), and datasets from C1, C1_HT, and ICELL_8 have 3 preprocessing methods with expected value of $\frac{1}{3}$ (red dashed line). The top row shows results for cancer cells (Cell Line A) by (A) FFNN and (B) XGB auditors. The bottom row shows results for normal cells (Cell Line B) by (C) FFNN and (D) XGB auditors. Audits are performed 100 times and results are averaged.

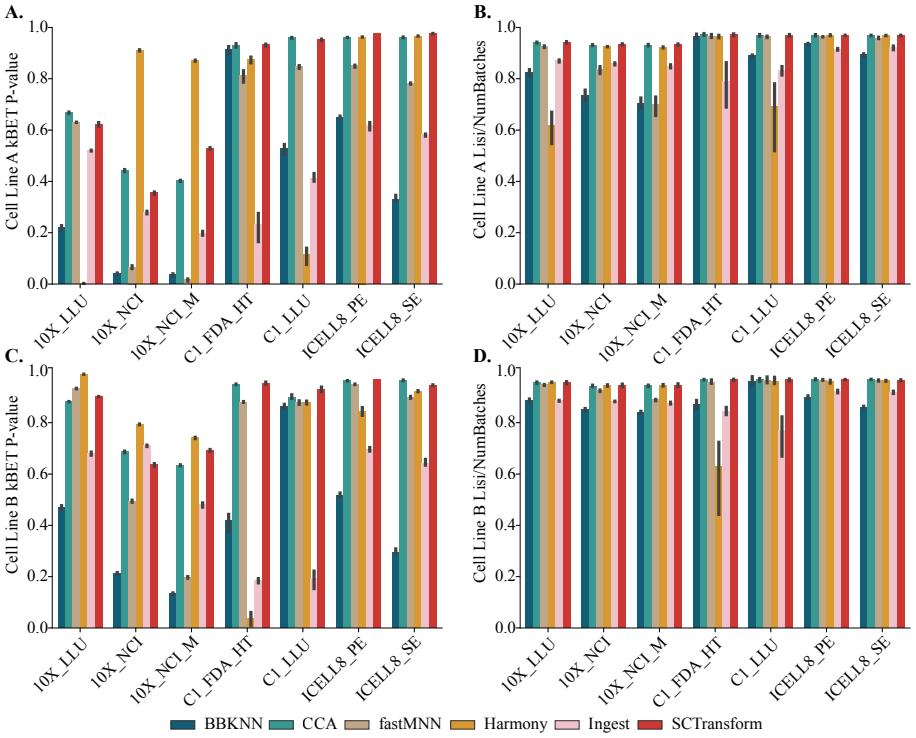

Figure 4: Quantitative evaluation of batch correction of 6 integration methods on 48 benchmark datasets. Shown are the bar plots of kBET P-Value scores (A, C) and normalized Lisi (B, D) of each integrated dataset from same centers and platforms but different preprocessing techniques. The top row shows results for cancer cells (Cell Line A). The bottom row shows results for normal cells (Cell Line B).

may be due to greater heterogeneity of cancer cells, which has been noted previously [9, 37]. This finding is also supported by the average kBET P-values and normalized Lisi scores of 0.57 and 0.89 for the cancer cell line compared with 0.69 and 0.92 for the normal cell line (Figure 4). As expected, in the integration of 10X Genomics datasets, larger uncorrected batch effects are observed for datasets preprocessed using umitools and zumi, while cellranger2.1 and cellranger3.0 datasets mixed well and have smaller residual batch effects.

Second, cells sequenced using C1 and C1_HT are more challenging to BKNN, Harmony and Ingest than cells sequenced using 10X Genomics and ICELL8, resulting in larger uncorrected batch effects in the integrated data. A possible explanation for this result is that integration methods have been optimized for 10X Genomics, which is more prevalent that the other 3 sequencing platforms. However, integration methods struggle to remove batch effects in some of the 10X Genomics datasets as well (Figure 3). Surprisingly, the integration results vary between sequencing centers, and cells sequenced at NCI are more challenging to integrate than cells sequenced at LLU. This finding is reproducible and consistent for cell lines sequenced individually (10X_NCI) and as mixtures (10X_NCI_M) at NCI.

Third, as previously reported, no single integration method outperforms others on all datasets. ML-IMS results agree with kBET and Lisi. For example, ML-IMS, kBET P-value, and normalized Lisi find that Harmony has the worst performance in C1_FDA_HT benchmark of cancer cell line, followed by BBKNN and Ingest. On the other hand, there are several disagreements between kBET and Lisi. For most datasets, normalized Lisi scores are high, indicating good integration. However, kBET P-value scores for some of these datasets are very low, in particular for 10X Genomics datasets (Figure 4).

The scrnabench benchmarks comprise cells of 2 well characterized cell lines sequenced and preprocessed in a variety ways. Therefore, they are suitable for the in-depth analysis of technical and biological variability encountered by different integration methods. We integrate all datasets into one using Harmony and perform ML-IMS audits of technical (centers, platforms, preprocessing and datasets) and biological variability. We use Harmony for computational efficiency. We find that technical variations were identifiable beyond random chance in all audits (Table 1). Technology has higher batch effects than the sequencing center and preprocessing methods and even the individual datasets can be detected in the integrated data. Cells of 2 cell lines are well-separated but overcorrected. Cell-line classification is 0.73 (FFNN) and 0.85 (XGB).

By default, ML-IMS works with 2-dimensional UMAP embeddings. The reason is because 2-dimensional UMAP embeddings are commonly used in visualizations and downstream cluster analyses.

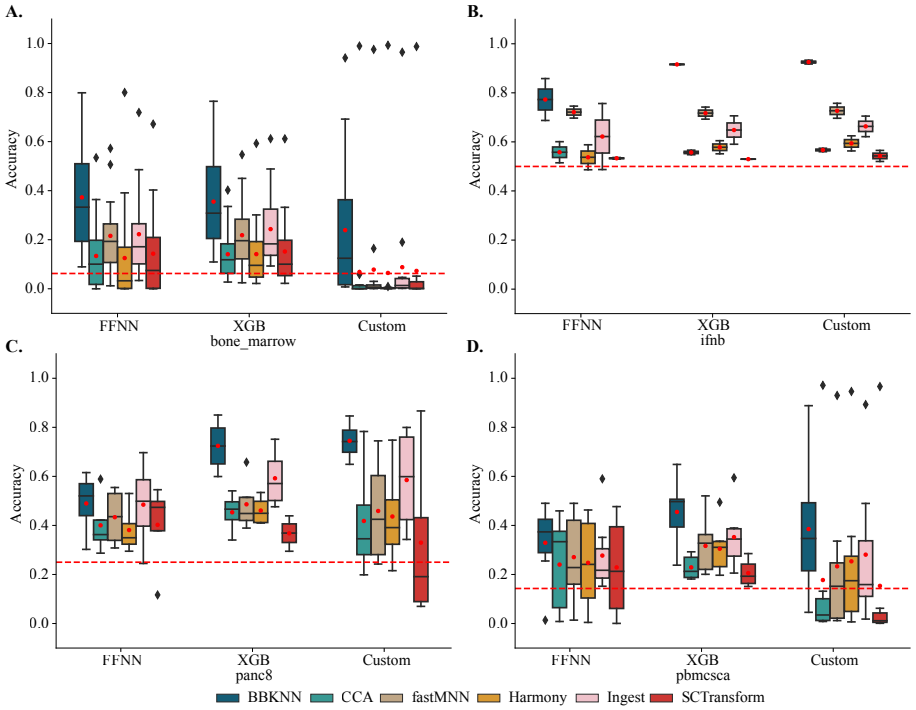

Figure 5: Audit of 6 integration methods on 4 popular benchmark datasets. Shown are the box plots of the distributions of classification accuracy scores averaged across all class labels of each integrated dataset: (A) bone_marrow, (B) ifnb, (C) panc8, and (D) pbmcsca. Audits are performed 100 times and results are averaged for FFNN, XGB and custom (KNN) auditors.

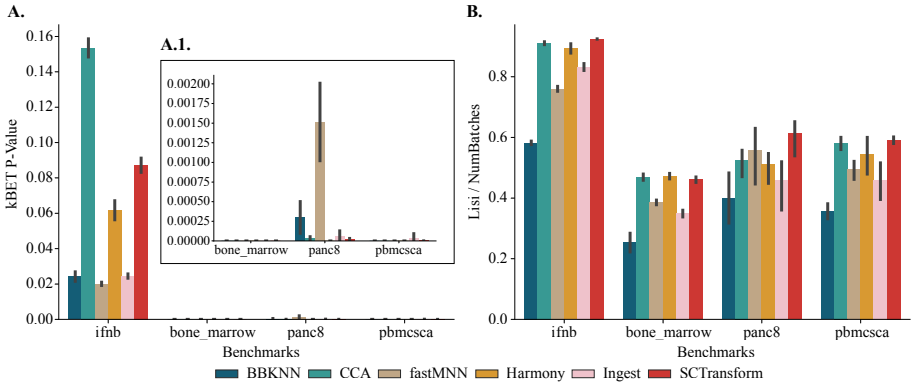

Figure 6: Quantitative evaluation of batch correction of 6 integration methods on 4 popular benchmark datasets. Shown are the bar plots of kBET P-Value (A) and normalized Lisi (B) of ifnb, bone_marrow, panc8, and pbmcsca. Inset (A.1) shows kBET P-Value scores of bone_marrow, panc8, and pbmcsca.

We examine how ML-IMS results change for different representation of Harmony-integrated and non-integrated data. Harmony performs principal component analysis (pca) first and then transforms 30 principal components into harmonized embeddings of size 30, followed by UMAP. Classification accuracy increases when 30 harmony embeddings are used to train auditors instead of UMAP (Table 1). When we apply UMAP directly to principal components and audit non-integrated data, we observe higher classification accuracy of batch effects than that of UMAP embeddings of harmonized data. However, cell-lines are better classified in non-integrated data and dimensionality reduction of Harmony embeddings using UMAP makes it harder to classify the biological variability.

We also validate ML-IMS on 4 benchmark datasets of past benchmarking studies, such as bone_marrow, ifnb, panc8 and pbmcsa. We confirm that no integration method performs best on all datasets (Figure 5). BBKNN and Ingest have greater uncorrected batch effects than other methods, while CCA, Harmony and SCTransform produce better results. Our audits reveal that integrated batches remain

Table 1: Audit of 24 10X Genomics datasets. For each data preparation type (rows), shown are the size and dimension of training and test subsets and average accuracy of classifying 4 sequencing centers, 4 sequencing technologies, 7 preprocessing methods, 48 datasets and 2 cell lines. Expected classification accuracy is shown in parentheses. HGV, PCA, PCA+UMAP report results for non-integrated data and PCA+HARMONY and PCA+HARMONY+UMAP report results of Harmony integration of 24 datasets.

	Size & dimension		Center (0.25)		Technology (0.25)		Preprocessing (0.14)		Dataset (0.02)		Cell line (0.5)	
Data Preparation	**Training**	**Test**	**FFNN**	**XGB**	**FFNN**	**XGB**	**FFNN**	**XGB**	**FFNN**	**XGB**	**FFNN**	**XGB**
HVG	(2268, 2000)	(630, 2000)	0.990	0.983	0.997	0.995	0.850	0.856	0.738	0.681	0.999	0.998
PCA	(2268, 30)	(630, 30)	0.993	0.982	0.998	0.994	0.834	0.810	0.658	0.551	0.999	0.997
PCA+UMAP	(2268, 2)	(630, 2)	0.801	0.944	0.939	0.996	0.471	0.628	0.259	0.411	0.951	0.998
PCA+HARMONY	(2268, 30)	(630, 30)	0.811	0.829	0.948	0.957	0.441	0.424	0.258	0.207	0.936	0.923
PCA+HARMONY+UMAP	(2268, 2)	(630, 2)	0.473	0.569	0.620	0.707	0.197	0.205	0.058	0.096	0.732	0.845

well discernible regardless of the integration method. The average classification accuracy of batches in each benchmark dataset and for each integration method is higher than expected. For example, in panc8 integration, both auditors flag celseq and smartseq2 batches as having the largest residual batch effects in all integrations, except for Harmony. In the integration of bone_marrow, auditors agree that batch effects are not fully corrected for MantonCB1, 3, and 8. In the pbmcsca integration, batches sequenced using inDrops, 10x Chromium (v2) A, Drop-seq, and 10x Chromium (v2) are not well-mixed with the other batches.

Absolute scores of kBET P-value and Lisi agree with these results, with some exceptions. First, based on kBET P-value, CCA has the best performance in ifnb integration, and fastMNN performs best in panc8 and pbmcsca integrations (Figure 6A). Due to its high sensitivity, kBET P-value generates 0 for all integration methods in the bone_marrow benchmark, making it impossible to determine the best integration method using this metric. On the other hand, normalized Lisi suggests that SCTransform is the best integration method for ifnb, panc8 and pbmcsca, and Harmony is the best integration method for bone_marrow (Figure 6B), reaching the same conclusion as the ML-IMS.

The greatest disagreement between ML-IMS and existing metrics is observed for the ifnb benchmark. This benchmark comprises 2 batches of control and stimulated cells. Relative to other benchmark datasets, kBET P-value and normalized Lisi scores are higher for ifnb, suggesting better batch effect correction. Both ML-IMS auditors, however, report residual batch effects for all integration methods, suggesting that differences between control and stimulated cells persist in the integrated data. This is supported by the qualitative evaluation as well. While the projection of the integrated data onto 2-dimensional UMAP embeddings shows good overlaps between control and stimulated cells, detailed visual inspections reveal that several areas contain cells from one batch only, which is picked by the ML-IMS auditors

We note that FFNN and XGB do not always agree in their audits. For instance, FFNN and XGB auditors fully agree on bone_marrow and ifnb. On the other hand, XGB detects larger uncorrected batch effects in panc8 and pbmcsa. Additionally, FFNN auditors suggest Harmony as the best integration method for all datasets, whereas XGB recommends Harmony for bone_marrow only, and SCTransform for the remaining datasets. We attribute the superiority of SCTransform to its being one of the most recent integration methods. However, it is the most compute-intensive method. Among 4 benchmark datasets, panc8 is the most challenging to integrate, while cells of pbmcsca are well mixed by most integration methods (Table 2). Better integration of pbmcsca benchmark with 7 batches compared to panc8 with 4 batches is not surprising because this benchmark is overused in the bioinformatics research. Several integration methods use it to tune their algorithms and hence, it is expected that their performance will be more accurate on this dataset.

Using ML-IMS programming interface, we added a third auditor, kNN, and repeated experiments. Our results show that this custom auditor fully agrees with XGB on ifnb. On 3 other benchmarks, kNN produces results similar to XGB but with lower estimates of uncorrected batch effects (Figure 5). To smooth possible disagreements between the different auditors, ML-IMS employs an ensemble approach to automatically select the best integration (Table 2).

One practical limitation of ML-IMS is the computational time of executing 100 audits. For example, the running time and maximum memory consumption of performing audits on ifnb, pbmcsca, and panc8 are 2.26 hours (4.8 Gigabytes), 1.63 hours (5.2 Gigabytes), and 1.14 hours (4.9 Gigabytes). However, we find that while 100 audits provide statistical power, in most practical applications, 10 audits provide similar results, thus, significantly reducing the compute time. In addition, ML-IMS package supports parallel execution.

5 CONCLUSION

This work presented a new method for the automated selection of the best integration of scRNA-seq datasets. ML-IMS repeatedly trains an ensemble of supervised machine learning classifiers that learn how to recognize the technical or biological variability of integrated cells in low dimensions. Motivated by the need of the end users to compare and contrast data integrated using existing and emergent techniques, ML-IMS can also be used in benchmarking studies to examine algorithmic biases of integration methods and to rank the performance of different tools. ML-IMS provides a convenient and interpretable threshold for the expected classification accuracy, and the package can be extended by adding, removing or substituting default auditors. Because ML-IMS is not distance-based, it can perform audits of integrated data of higher dimensions and it does not suffer from the curse of dimensionality. By employing ML-IMS, researchers can save time and effort in selecting suitable integration methods for their specific datasets, thereby improving the efficiency and quality of scRNA-seq data downstream analysis.

Table 2: IMS scores of different ensembles of auditors. Shown are the IMS scores of 4 benchmarks integrated using 6 methods. Different ensembles of auditors, include FFNN (F), XGB (X), and KNN (K). The best integration selected by each ensemble is shown in boldface.

	Integrations	F	X	K	F+X	F+K	X+K	F+X+K
bone_marrow	CCA	0.069	0.079	0.006	0.074	0.038	0.042	0.052
	SCTransform	0.081	0.089	0.011	0.086	0.045	0.050	0.060
	Harmony	**0.062**	**0.079**	**0.002**	**0.071**	**0.032**	**0.040**	**0.048**
	fastMNN	0.155	0.156	0.015	0.155	0.085	0.086	0.109
	BBKNN	0.309	0.284	0.176	0.301	0.243	0.238	0.258
	Ingest	0.158	0.179	0.025	0.170	0.092	0.103	0.122
ifnb	CCA	0.059	0.057	0.068	0.059	0.057	0.060	0.060
	SCTransform	0.032	**0.027**	**0.044**	**0.033**	**0.039**	**0.033**	**0.035**
	Harmony	**0.039**	0.080	0.099	0.057	0.066	0.083	0.067
	fastMNN	0.220	0.216	0.225	0.219	0.226	0.226	0.224
	BBKNN	0.274	0.411	0.428	0.344	0.350	0.421	0.371
	Ingest	0.121	0.144	0.164	0.135	0.140	0.149	0.146
pancreas	CCA	0.168	0.200	0.161	0.178	0.152	0.179	0.182
	SCTransform	0.143	**0.134**	**0.092**	**0.138**	**0.124**	**0.102**	**0.114**
	Harmony	**0.124**	0.190	0.180	0.170	0.165	0.188	0.169
	fastMNN	0.193	0.218	0.202	0.212	0.189	0.213	0.213
	BBKNN	0.235	0.471	0.490	0.357	0.359	0.492	0.400
	Ingest	0.239	0.371	0.350	0.289	0.290	0.342	0.306
pbmcsca	CCA	0.091	0.082	0.025	0.086	0.063	0.050	0.066
	SCTransform	0.093	**0.068**	**0.011**	**0.073**	**0.050**	**0.040**	**0.055**
	Harmony	**0.089**	0.160	0.109	0.139	0.111	0.134	0.128
	fastMNN	0.120	0.166	0.098	0.148	0.109	0.135	0.132
	BBKNN	0.192	0.313	0.244	0.253	0.228	0.300	0.243
	Ingest	0.136	0.210	0.135	0.164	0.125	0.171	0.158

In the future, we plan to implement an additional functionality which will simultaneously perform audits of technical and biological variability using multi-label classification. The package will also be ported to the R programming language and incorporated into the scrnabench package.

ACKNOWLEDGEMENTS

This research was partially supported by the Pilot Award from the Wake Forest Center for Biomedical Informatics. The authors acknowledge the Distributed Environment for Academic Computing (DEAC) at Wake Forest University for providing HPC resources that have contributed to the research results reported within this paper.

REFERENCES

[1] Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/
[2] Elham Azizi et al. 2018. Single-Cell Map of Diverse Immune Phenotypes in the Breast Tumor Microenvironment. *Cell* 174, 5 (Aug 2018), 1293–1308.e36.
[3] Sapan Bhandari et al. 2022. Multi-Target Integration and Annotation of Single-Cell RNA-Sequencing Data. In *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics* (Northbrook, Illinois) *(BCB '22)*. Association for Computing Machinery, New York, NY, USA, Article 29, 4 pages.
[4] Andrew Butler et al. 2018. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology* 36, 55 (May 2018), 411–420.
[5] Maren Büttner et al. 2019. A test metric for assessing single-cell RNA-seq batch correction. *Nature methods* 16, 1 (2019), 43–49.
[6] Ruben Chazarra-Gil et al. 2021. Flexible comparison of batch correction methods for single-cell RNA-seq using BatchBench. 49 (Feb 2021), e42.
[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794.
[8] Wanqiu Chen et al. 2021. A multicenter study benchmarking single-cell RNA sequencing technologies using reference samples. *Nature Biotechnology* 39, 9 (2021), 1103–1114.
[9] Xin Chen et al. 2021. A multi-center cross-platform single-cell RNA sequencing reference dataset. *Scientific data* 8, 1 (2021), 1–11.
[10] Jiarui Ding et al. 2020. Systematic comparison of single-cell and single-nucleus RNA-sequencing methods. *Nature Biotechnology* 38, 6 (Jun 2020), 737–746.
[11] Dominic Grün et al. 2016. De Novo Prediction of Stem Cell Identity using Single-Cell Transcriptome Data. *Cell Stem Cell* 19, 2 (Aug 2016), 266–277.
[12] Christoph Hafemeister and Rahul Satija. 2019. Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biology* 20, 1 (Dec 2019), 296.
[13] Laleh Haghverdi et al. 2018. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nature Biotechnology* 36, 55 (May 2018), 421–427.
[14] Stephanie C Hicks et al. 2018. Missing data and technical variability in single-cell RNA-sequencing experiments. *Biostatistics* 19, 4 (Oct 2018), 562–578.
[15] Dragomirka Jovic et al. 2022. Single-cell RNA sequencing technologies and applications: A brief overview. *Clinical and Translational Medicine* 12, 3 (2022), e694.
[16] Hyun Min Kang et al. 2018. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nature Biotechnology* 36, 11 (Jan 2018), 89–94.
[17] Ilya Korsunsky et al. 2019. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature Methods* 16, 1212 (Dec 2019), 1289–1296.
[18] Ilya Korsunsky et al. 2023. Methods to compute Local Inverse Simpson's Index (LISI). https://github.com/immunogenomics/LISI
[19] Malte D Luecken et al. 2022. Benchmarking atlas-level data integration in single-cell genomics. *Nature methods* 19, 1 (2022), 41–50.
[20] Feiyang Ma and Matteo Pellegrini. 2020. ACTINN: automated identification of cell types in single cell RNA sequencing. *Bioinformatics* 36, 2 (2020), 533–538.
[21] Evan Z. Macosko et al. 2015. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell* 161, 5 (May 2015), 1202–1214.
[22] F. Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[23] Krzysztof Polański et al. 2020. BBKNN: fast batch alignment of single cell transcriptomes. *Bioinformatics* 36, 3 (Feb 2020), 964–965.
[24] Ren Qi et al. 2020. Clustering and classification methods for single-cell RNA-sequencing data. *Briefings in bioinformatics* 21, 4 (2020), 1196–1208.
[25] Aviv Regev et al. 2017. The human cell atlas. *elife* 6 (2017), e27041.
[26] Satijalab. 2023. Dataset distribution for Seurat. https://github.com/satijalab/seurat-data
[27] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (Jul 1948), 379–423.
[28] Tim Stuart et al. 2019. Comprehensive Integration of Single-Cell Data. *Cell* 177, 7 (Jun 2019), 1888–1902.e21.
[29] Tim Stuart and Rahul Satija. 2019. Integrative single-cell analysis. *Nature Reviews Genetics* 20, 55 (May 2019), 257–272.
[30] Valentine Svensson, Roser Vento-Tormo, and Sarah A. Teichmann. 2018. Exponential scaling of single-cell RNA-seq in the past decade. *Nature Protocols* 13, 44 (Apr 2018), 599–604.
[31] Yuqi Tan and Patrick Cahan. 2019. SingleCellNet: a computational tool to classify single cell RNA-Seq data across platforms and across species. *Cell systems* 9, 2 (2019), 207–213.
[32] Antonio Torralba and Alexei A. Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. 1521–1528.
[33] Hoa Thi Nhu Tran et al. 2020. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biology* 21, 1 (Jan 2020), 12.
[34] Po-Yuan Tung et al. 2017. Batch effects and the effective design of single-cell gene expression studies. *Scientific Reports* 7, 11 (Jan 2017), 39921.
[35] Nathan Whitener and Konghao Zhao. 2023. Scrnabench: A package for metamorphic benchmarking of scRNA-seq data analysis methods. https://github.com/NWhitener/scrnabench
[36] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. 2018. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology* 19, 1 (Feb 2018), 15.
[37] Konghao Zhao, Jason M. Grayson, and Natalia Khuri. 2023. Multi-Objective Genetic Algorithm for Cluster Analysis of Single-Cell Transcriptomes. *Journal of Personalized Medicine* 13, 2 (2023).
[38] Grace X. Y. Zheng et al. 2017. Massively parallel digital transcriptional profiling of single cells. *Nature Communications* 8, 11 (Jan 2017), 14049.
[39] Bin Zou et al. 2021. deepMNN: Deep Learning-Based Single-Cell RNA Sequencing Data Batch Correction Using Mutual Nearest Neighbors. *Frontiers in Genetics* 12 (2021).