Published online 1 February 2021 *Nucleic Acids Research, 2021, Vol. 49, No. 7* ***e42***
doi: 10.1093/nar/gkab004

Flexible comparison of batch correction methods for single-cell RNA-seq using BatchBench

Ruben Chazarra-Gil*, Stijn van Dongen, Vladimir Yu Kiselev* and Martin Hemberg*

Wellcome Sanger Institute, Wellcome Genome Campus, Hinxton CB10 1SA, UK

Received May 28, 2020; Revised December 11, 2020; Editorial Decision December 28, 2020; Accepted January 29, 2021

ABSTRACT

As the cost of single-cell RNA-seq experiments has decreased, an increasing number of datasets are now available. Combining newly generated and publicly accessible datasets is challenging due to non-biological signals, commonly known as batch effects. Although there are several computational methods available that can remove batch effects, evaluating which method performs best is not straightforward. Here, we present BatchBench (https://github.com/cellgeni/batchbench), a modular and flexible pipeline for comparing batch correction methods for single-cell RNA-seq data. We apply BatchBench to eight methods, highlighting their methodological differences and assess their performance and computational requirements through a compendium of well-studied datasets. This systematic comparison guides users in the choice of batch correction tool, and the pipeline makes it easy to evaluate other datasets.

INTRODUCTION

Single-cell RNA sequencing (scRNA-seq) technologies have made it possible to address biological questions that were not accessible using bulk RNA sequencing (1), e.g. identification of rare cell types (2,3), discovery of developmental trajectories (4–6), characterization of the variability in splicing (7–11), investigations into allele specific expression (12–15) and analysis of stochastic gene expression and transcriptional kinetics (11,16). There are currently a plethora of different protocols and experimental platforms available (17,18). Considerable differences exist among scRNA-seq protocols with regards to mRNA capture efficiency, transcript coverage, strand specificity, UMI inclusion and other potential biases (17,18). It is well known that these and other technical differences can impact the observed expression values, and if not properly accounted for they could be confounded with biological signals (19). Such differences arising due to non-biological factors are commonly known as batch effects.

Fortunately, with appropriate experimental design it is possible to remove a portion of the batch effects computationally, and recently there has been a large degree of interest in developing such methods for scRNA-seq. We group the methods into three categories depending on what space they operate on with respect to the expression matrix (Figure 1A). The expression matrix represents the number of reads found for each cell and gene, and it is central to computational analyses. The first set of methods, mnnCorrect, limma, ComBat, Seurat 3 (hereafter referred to as Seurat) and Scanorama, produce a merged, corrected expression matrix. The second set, Harmony and fastMNN, instead operate on a low-dimensional embedding of the original expression matrices. As such their output cannot be used for downstream analyses which require the expression matrix, limiting their use for some applications. Finally, the BBKNN method operates on the *k*-nearest neighbor graph constructed from the expression matrices and consequently its output is restricted to downstream analyses where only the cell label can be used.

As the choice of batch correction method may impact the downstream analyses, the decision of which one to use can be consequential. To decide what method to use, most researchers rely on benchmarking studies. Traditionally such comparisons are carried out using a compendium of relevant datasets. The downside of this approach is that methods published after the benchmark was carried out are not included and that the comparison may not have featured datasets that contain all the relevant features required to evaluate the methods. To overcome these issues we have developed BatchBench (Figure 1B), a flexible computational pipeline which makes it easy to compare both new methods and datasets using a variety of criteria. Here we report on the comparison of eight popular batch effect removal methods (Table 1) using three well-studied scRNA-seq datasets. BatchBench is implemented in Nextflow (20) and it is freely available at https://github.com/cellgeni/batchbench under the MIT Licence.

*To whom correspondence should be addressed. Tel: +44 1223499955; Email: rubenchazarra@gmail.com
Correspondence may also be addressed to Vladimir Yu Kiselev. Email: vk6@sanger.ac.uk
Correspondence may also be addressed to Martin Hemberg. Email: mh26@sanger.ac.uk

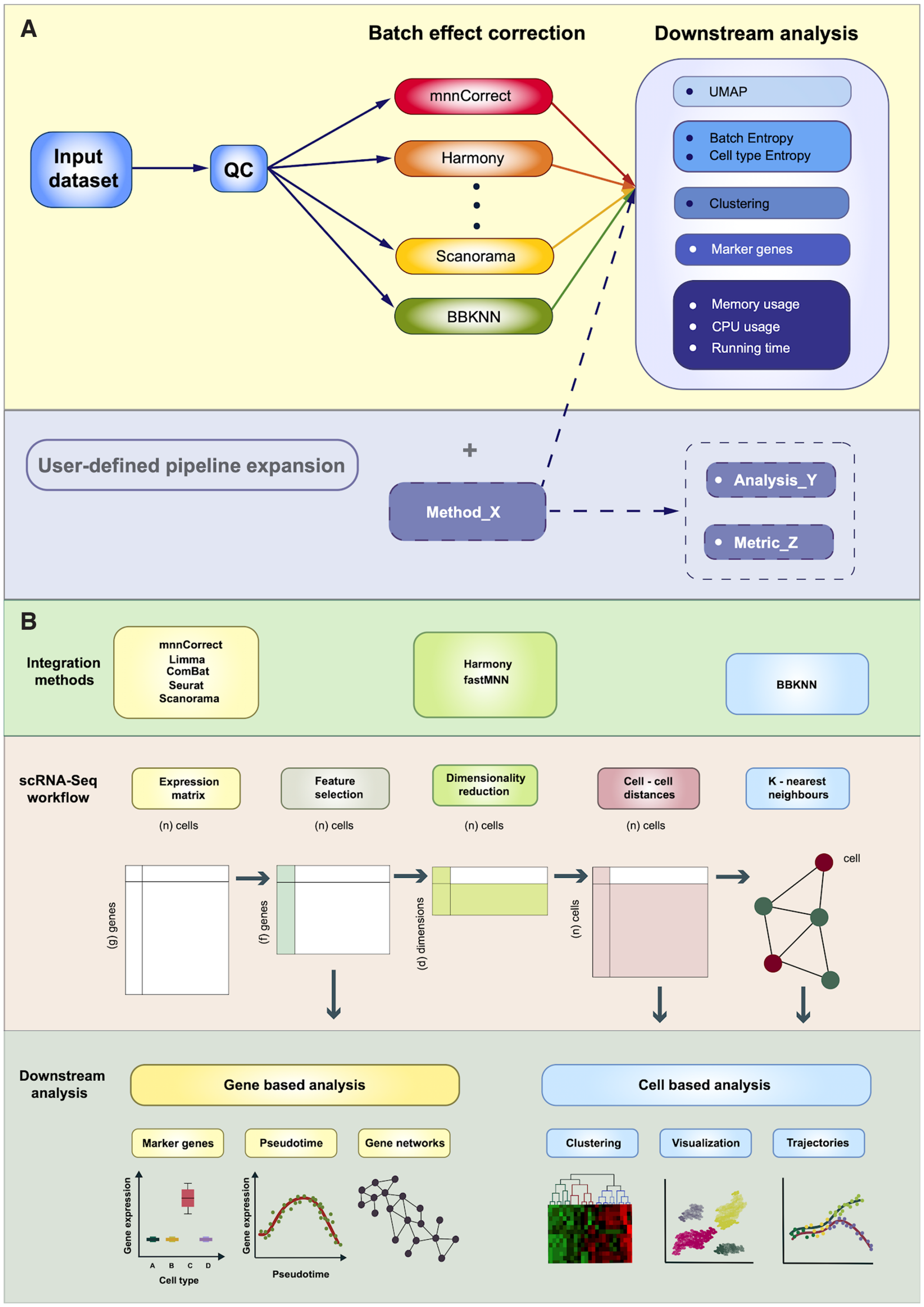

Figure 1. Overview of the BatchBench pipeline workflow and schematic representation of the conventional scRNA-seq data analysis pipeline from the expression matrix. (**A**) Batchbench first carries out QC on the input dataset prior to performing batch correction with the eight methods selected. After this, a series of downstream analyses are computed, including: UMAP coordinates, Shannon entropies, clustering and marker gene analysis, and resource consumption metrics of each of the processes. (**B**) Central and lower panels depict the conventional scRNA-seq data analysis pipeline and the analyses that can be carried out with the output of each step. Upper panel represents the space over which each of the batch correction methods operate. The initial expression matrix typically undergoes feature selection, being then source for gene based analyses, as marker gene and pseudotime analysis or gene networks. Methods mnnCorrect, Limma, ComBat, Seurat and Scanorama operate in the expression matrix space. Next, a dimensionality reduction step is performed. Methods Harmony and fastMNN operate in this space. The low dimensional embedding is then converted into a matrix of cell-cell distances which in turn can be converted to a graph. These are inputs for cell based analysis as clustering, visualization and trajectory inference of cells. BBKNN method operates in this graph space.

Table 1. Summary of the eight batch correction methods considered in this study. Programming language of the method, type of output object, tool's batch correction principle as well as installation source and license type are listed

Tool	Lang.	Output	Correction principle	Installation	License	Ref
mnnCorrect	R	Counts matrix	Mutual nearest neighbour detection across batches.	Batchelor ♪ ○	GPL-3	(16)
Limma	R	Counts matrix	Fits linear model to remove batch effect components.	Limma ♪ ○	GPL (>=2)	(21)
ComBat	R	Counts matrix	Adjusts for known batches using an empirical Bayesian framework.	Sva ♪ ○	Artistic-2.0	(22)
Seurat	R	Counts matrix	Diagonalized CCA to reduce dimensionality and MNN detection in this space.	Seurat (CRAN) ○	GPL-3	(23)
Scanorama	🐍	Counts matrix	SVM to reduce dimensionality and mutual nearest neighbor detection and panoramic stitching.	pip ○	MIT	(24)
Harmony	R	Embedding	Iterative soft k-means clustering algorithm in dimensionally reduced space.	Github ♪ ○	GPL-3	(25)
fastMNN	R	Embedding	Mutual nearest neighbor detection after multi-sample PCA.	Batchelor ○	GPL-3	(16)
BBKNN	🐍	Graph	Mutual nearest neighbour pair selection across batches in PCA space.	pip3 ○	MIT	(26)

R **R,** ♪ **Bioconductor,** ○ **Conda,** 🐍 **Python.**

By default, BatchBench evaluates batch correction methods based on two different entropy metrics. The normalized Shannon entropy is used to quantify how well batches are aligned while preserving the separation of different cell populations. However, the entropy measures do not provide a complete picture of how the batch correction impacts downstream analyses. Therefore, BatchBench has a modular design to allow users to incorporate additional metrics, and we provide two examples of such metrics - unsupervised clustering and identification of marker genes. Five different unsupervised clustering methods are applied to the merged cells to afford the user a better understanding of how the different methods affect this step which is often central to the analysis. We also compare cell-type specific marker genes to understand how different batch correction methods affect the expression levels.

METHODS

Datasets

Pancreas dataset. We consider three published pancreas datasets: Baron (GSE84133) (39), Muraro (GSE85241) (27) and Segerstolpe (E-MTAB-5061) (28) generated using inDrop, CEL-Seq2 and Smart-Seq2 technologies, respectively. Initially, quality control was performed on each of the datasets to remove cells with <200 counts and genes that were present in <3 cells along with spike-ins and antisense transcripts. Furthermore, we only retained cells that had been assigned a biologically meaningful cell type (e.g. removing cells from the 'unclassified' category).

For Figure 3, we wanted to represent the pancreas results as a boxplot similar to the other datasets. To ensure that we got a distribution we considered three additional versions of the data. One of these versions contained all of the genes expressed across the three batches rather than just the highly variable ones. The second contained 1000 cells selected randomly from each batch using the highly variable genes. The third version contained only six cell types (acinar, alpha, beta, gamma, delta and ductal) from each batch downsampled to 50% of the original number of cells and information from the highly variable genes.

Mouse cell atlas datasets. Individual MCA datasets were downloaded from the paper's Figshare site and merged by tissue, generating 37 organ datasets. From these, 18 datasets containing more than one batch and with a reasonable proportion of cells across batches were selected. Through further preprocessing we removed cells expressing <250 genes, genes expressed in <50 cells, cell types representing <1% of total cell population in a tissue, and batches containing <5% of the total number of cells in a tissue (Supplementary Table S2).

Tabula muris datasets. The data was downloaded from the paper's Figshare site. For all analyses except Figure 4, individual datasets representing the same tissue across the two platforms were merged into 11 organ datasets (Supplementary Table S1). We set workflow quality control parameters to remove cells expressing <1000 genes, genes expressed in <50 cells. Again, cell types representing <1% of total cell population in a tissue, and batches containing <5% of the total number of cells in a tissue were excluded from further analyses. For the scaling analysis in Figure 4, the previous tissues were merged into an atlas Tabula Muris dataset which was filtered to retain cells with >200 genes expressed, genes expressed in >3 cells. Cells assigned to NA or unknown cell types were excluded. Cell types representing <1% of total cell population in a tissue, and batches containing <5% of the total number of cells in a tissue were excluded from further analyses. This resulted in an object of 4168 genes and 60 828 cells (40 058 from *10X* and 20 770 from *Smart-Seq2*).

Batch and cell type entropy

The output of each tool is transformed into a K Nearest Neighbour graph with each node i representing a cell (`BuildKNNGraph`, scran package). Each cell is connected to its $k = 30$ nearest neighbors as defined by the similarity of expression profiles calculated using the Euclidean distance. Using the graph, we calculate for each cell i the probability that a neighbor has cell type c, P_{ic}, as well as the probability that a neighbor comes from batch b, P_{ib}. From these joint probabilities we can calculate cell type and batch entropies. We report the average value across all cells divided by the theoretical maximum to ensure a value in the interval [0, 1]. For the datasets considered in this study, the results are robust with respect to the choice of k (Supplementary Figure S16).

UMAP

Uniform Manifold Approximation and Projection (UMAP) is computed through the `scanpy.api.tl.umap` function, which uses the implementation of umap-learn (38). For the batch removal methods implemented in R, the rds objects are first converted into h5ad objects using the `sce2anndata` from the sceasy package (https://github.com/cellgeni/sceasy/).

Downsampling

The filtered Tabula Muris dataset was sampled using uniform selection and no replacement to 1, 2, 5, 10, 20 and 50% of its cells. Resulting in objects of: 4168 genes and 608, 1217, 3041, 6083, 12 166 and 30 414 cells. The initial proportion of the batches (0.64, 0.36) was maintained through the different subsets.

Artificial batches

We work with a reduced version of the Tabula Muris atlas object. We first removed all the Smart-seq2 cells and then retained only the 10 largest cell types. From this 1001 cells are randomly sampled to serve as input to the artificial batch generation. All 4168 initial genes are considered. We base our simulation of batch effects on a normal distribution. For each batch to be simulated, we define: (i) a fraction f of cells sampled with uniform probability from the sequence [0.05, 0.1, 0.15, … 1.0]; (ii) a value d representing the dispersion of the effect to be simulated sampled with uniform probability from the sequence [0.5, 1.0, 1.5, … n], where n is the number of batches to simulate. For each of the 10 cell types in the input data we add count values by drawing values from a normal distribution with a standard deviation d. The artificial batch effect is only applied to those genes expressed in $>f$ of the cells. If a gene is assigned a negative value, then it is replaced by 0. The result is a simulated data set of 1001 cells and 4168 genes which is appended to the input data set. We followed this approach to simulate data sets with 2, 3, 5, 10, 20 and 50 equally sized batches.

Feature selection

We rank genes in descending order by their coefficient of variation establishing five fractions of features: 0.05, 0.1, 0.2, 0.5 and 1.0 (all of the features). Feature selection is performed as a first step in each clustering algorithm script prior to any processing of the input data.

Clustering analysis

The merged samples were clustered using five different clustering algorithms: SC3 (35) from the homonim Bioconductor package, Louvain and Leiden as implemented in Seurat (23), RaceID (2) and standard hierarchical clustering using Ward's agglomeration method. SC3 and Race ID require a count matrix as input. For SC3 we set k to the number of cell populations of each dataset. If the dataset had >5000 cells we enable `sc3_run_svm` to speed up the processing. RaceID uses Euclidean distances based on the Pearson correlation distance. All three RaceID clustering options (k-means, k-medoids and hclust) are implemented in our clustering step. The other clustering algorithms can be applied to all batch correction methods in our study. Louvain and Leiden methods were implemented with the Seurat function `FindClusters`, with other parameters set to their default values. We also implemented standard hierarchical clustering using Ward's agglomeration method with the *hclust* function from the *stats* package.

As a pre-processing step after feature selection and prior to SC3, RaceID and hierarchical clustering, all cells and features with zero variance are removed. Moreover, in the case of RaceID clustering, negative values in the expression matrix that may result from the batch effect removal step are set to to zero.

To assess the similarity of each corrected output clustering annotation with the provided ground truth Adjusted Rand Index and Variation of Information are computed with the *arandi* and the *vi.dist* functions from the *mclust package respectively.*

To compare the results across datasets we select the best similarity metric value across all feature selection fractions considered (Figure 5, Supplementary Figure S12). Since variation of information is a distance metric, we perform a min–max normalization to scale the data across datasets (Supplementary Figure S12). For both metrics the feature fraction matching the best value is stored (Supplementary Figures S10 and S13). Additionally, we examine the correlation (as Pearson's ρ) between the similarity values and the feature range for which they were obtained (Supplementary Figures S11 and S14).

Marker gene analysis

To obtain marker genes we use the `FindMarkers` function from the Seurat package which restricts the comparison to methods that output a normalised count matrix. For a gene to be considered as a marker, we require that the absolute value of the log fold-change >2, and that the gene is expressed in at least half of the cells in each population. We use the default Wilcoxon Rank Sum test to find genes that are significantly different (adjusted P-value < 0.05) between the merged dataset, and in each of the individual batches.

To compare the overlap of the sets of marker genes identified across batches and the merged data we used the multiple site generalized Jaccard index (36). We restricted the comparison to the cell populations that are common to all individual batches. We also investigate the proportion of cell populations of the dataset for which marker genes can be found.

BatchBench pipeline

As an input, BatchBench (https://github.com/cellgeni/batchbench) requires a `SingleCellExperiment` (R based) or `AnnData` object (python based). The input object is then converted into its counterpart in the other language. This object must contain: log-normalized counts, and the batch and cell type annotation of their cells as `Batch` and `cell_type1` respectively, in the object metadata. The workflow performs an initial QC step where cells, genes, batches or cell types can be filtered according to user-defined parameters. Cells not assigned to any batch or cell type are excluded in this step also. Each dataset is then sent in parallel as input to each of the batch effect correction tools, after which rds and h5ad objects containing the output are saved and made available for the user. Each of the batch corrected outputs serves as input for a series of downstream analyses: (i) UMAP coordinates are computed and saved as a csv file for visualization of the different batch corrections, (ii) entropy computation and saved as csv file, (iii) clustering analysis, (iv) marker gene analysis and any module optionally added by the user.

RESULTS

Entropy measures quantify integration of batches and separation of cell types

To illustrate the use of BatchBench we first considered three scRNA-seq studies of the human pancreas (27–29). Even though the samples were collected, processed and annotated independently, several comparisons have shown that batch effects can be overcome (19,30). Visualization of the uncorrected data using UMAP reveals a clear separation of the major cell types across batches (Figure 2A). As expected, all of the methods in our study were able to merge equivalent cell populations from different batches while ensuring their separation from other cell types. Visual inspection suggests that Seurat and Harmony achieve groupings mainly driven by the cell types, whereas the other methods tend to aggregate the different batches. It is notable that BBKNN brings cell populations closer but is unable to superimpose the batches.

To evaluate how well the batch correction methods mix cells from different batches while keeping cell types separate, we computed the normalized Shannon entropy (16,29) based on the batch and cell type annotations provided by the original authors (Methods). The desired outcome is a high batch entropy, indicating a homogeneous mixture of the batches, and a low cell type entropy, suggesting that cell populations remain distinct. While all the methods were able to keep the distinct cell populations separate, we observed greater differences for the batch entropy (Figure 3). Based on this metric we consider Seurat and Harmony as the best methods. As intermediate performers Scanorama and fastMNN show a wider distribution of batch entropy values. Finally, mnnCorrect, Limma and ComBat can be considered the poorer performers in aligning the different batches.

We carried out similar investigations for the Mouse Cell Atlas (MCA) (31) and Tabula Muris (32) datasets. In the MCA the batches correspond to the eight different animals (31), and as the mice all come from the same genetic background and were raised in the same environment we expect the batch effects to be smaller than for the pancreas data. The batch entropy for the uncorrected data is indeed higher than for the pancreas data (Figure 3), and most methods are able to mix the batches of the MCA better, as confirmed by visual inspection. The cell type entropies are higher than for the pancreas data, and we hypothesize that this is a consequence of the fine-grained annotation which makes it difficult to separate cell types. For example, the bone marrow contains six different types of neutrophils and the testes five types of spermatocytes. Overall across MCA data, Seurat and Harmony show the best batch mixing, although at the cost of slightly increasing cell type mixing compared to the uncorrected counts and the other methods. Scanorama can also be considered a good performer followed by fastMNN.

Next, we investigated another mouse cell atlas, Tabula Muris (32) and our analysis shows a greater sample effect as evidenced by a very low batch entropy for the uncorrected data (Figure 3). Since the batches correspond to two different experimental platforms (32), it is not surprising that there are larger differences than for the MCA. Furthermore, all methods perform better with regards to the cell type entropy, potentially due to a more coherent annotation. For all three datasets, we note that for most methods there is greater variation in batch entropy than cell type entropy. Closer inspection reveals that the batch entropies vary substantially across tissues (Supplementary Table S1). Interestingly, all methods, except for Seurat and BBKNN, are unable to achieve high batch entropy for datasets with a small number of cell types. Closer inspection reveals that all methods except Seurat and BBKNN show a significant correlation between cell type entropy and number of cell types, suggesting poorer performance with more fine-grained annotation (Supplementary Figure S1). Taken together, Seurat consistently succeeds in mixing the batches, again at the cost of a slightly distinct cell population mixing. Scanorama performs well although with higher variation across datasets.

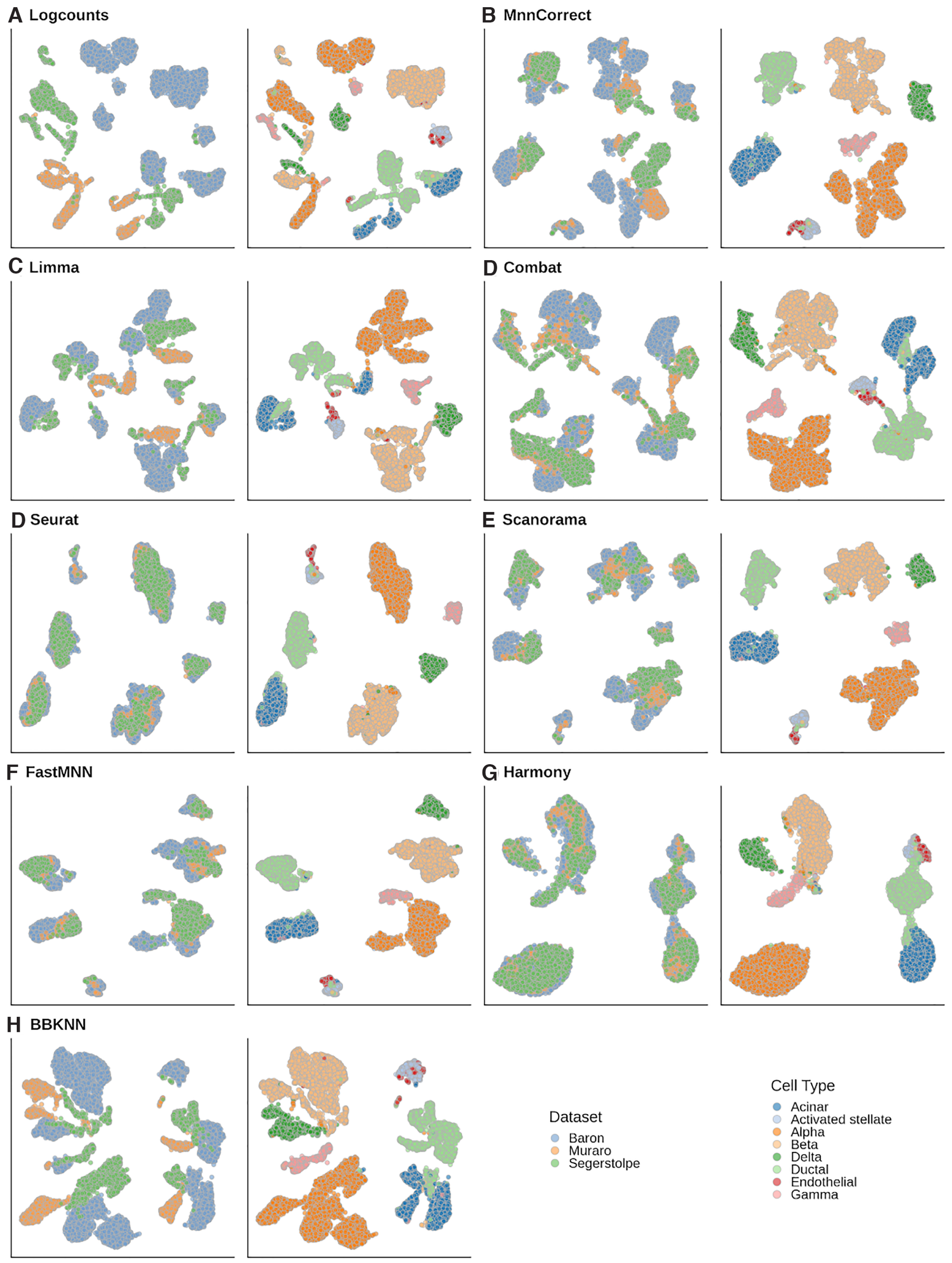

Figure 2. UMAP visualization of the human pancreas dataset prior and after batch correction with the eight different methods considered. (**A**) Original uncorrected data. (**B–H**) Corrected data. Each pair of panels shows the cells labeled either by dataset of origin (left) or cell type (right). A good batch correction should ensure that cells from different batches are grouped together while cells from distinct cell populations are retained separate.

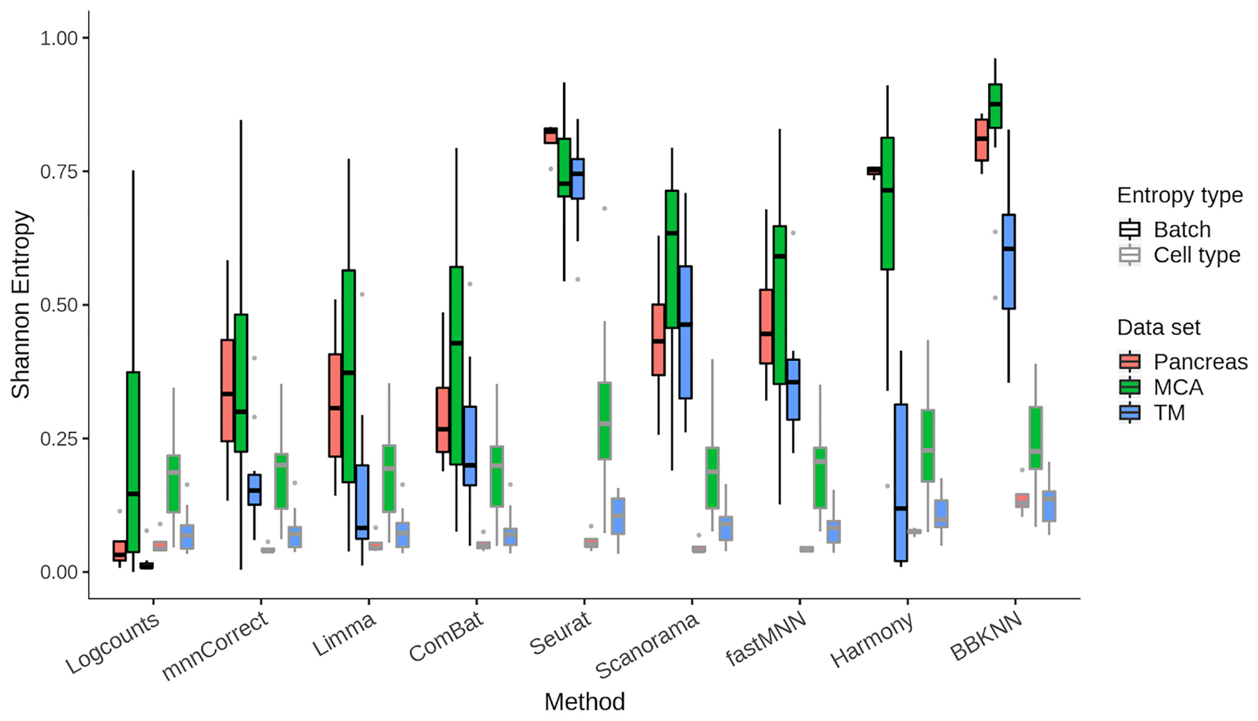

Figure 3. Batch and cell type entropies prior and after batch correction with the eight different methods considered. The boxplots show the Shannon entropy over batch (black) and cell type (gray) of the different batch effect correction methods for pancreas data (red), Mouse Cell Atlas (green) and Tabula Muris (blue). The black line represents the mean across the cells, the box the upper and lower quartiles, the whiskers 95th percentiles and the dots show outliers.

Surprisingly, Harmony is unable to properly align the Tabula Muris batches.

Batch correction becomes harder as the number of cells and the number of batches increase

To determine how the number of cells in each sample influences batch correction performance and running times we considered the Tabula Muris dataset, and downsampled it to 1, 5, 10, 20 and 50% of the original 60 828 cells (Methods). Across all subsets, the input objects contain 64% of 10X cells and 36% of FACS-sorted Smart-Seq2 cells. Note that this batch correction task is more challenging than the one in Figure 3 as we now merge cells from different tissues.

The number of cells has a strong impact on performance and it becomes more difficult to align the two batches with increasing cell numbers. All methods except Scanorama, Harmony and Seurat reduce the batch entropy by >50% as the number of cells increases from 608 to 60 828 (Figure 4A). Unfortunately, Scanorama mixes the cell types as well as batches, and surprisingly none of the entropies change as the number of cells increases. Harmony is the only method that, after an initial drop, increases the batch entropy with the number of cells. For all methods except Scanorama, the cell type entropy is also reduced, suggesting that it becomes easier to group cells from the same origin for larger datasets. With the exception of Scanorama, the majority of the methods do not significantly increase the cell type entropy above the value of the uncorrected counts, even decreasing it for the smaller subsets.

The main goal of the investigation involving different numbers of cells is to learn how the computational resource requirements change as this is an important factor when choosing a method. Considering the time required to perform the integration, we found substantial differences as ComBat, Limma, Harmony and BBKNN have more or less constant run times as the number of cells grow. By contrast, mnnCorrect and fastMNN grow exponentially, with the former being the slowest method in our study. Seurat initially has a stable runtime before it starts to grow exponentially (Figure 4B). For all methods we found that memory usage increases exponentially with the number of cells. The differences are smaller than for the run-time, with Seurat, mnnCorrect, ComBat and fastMNN consuming the most resources, while Harmony, Scanorama and BBKNN have the lowest requirements (Figure 4C). The memory requirements and runtimes observed in the scaling experiments are similar to what we found for the previous section (Supplementary Figure S2).

As sequencing costs decrease, the number of different samples that can be processed will increase. Thus, we also evaluated how well each method handles an increasing number of batches. For this study we considered subsets of the Tabula Muris 10X dataset with 4168 genes and 18 347 cells. As the batches created by subsampling this dataset are entirely artificial, we added small batch-specific random counts to each gene to ensure that there are differences that require correction (Methods). In our simulations, cell types are well separated whereas the batches are more overlapping.

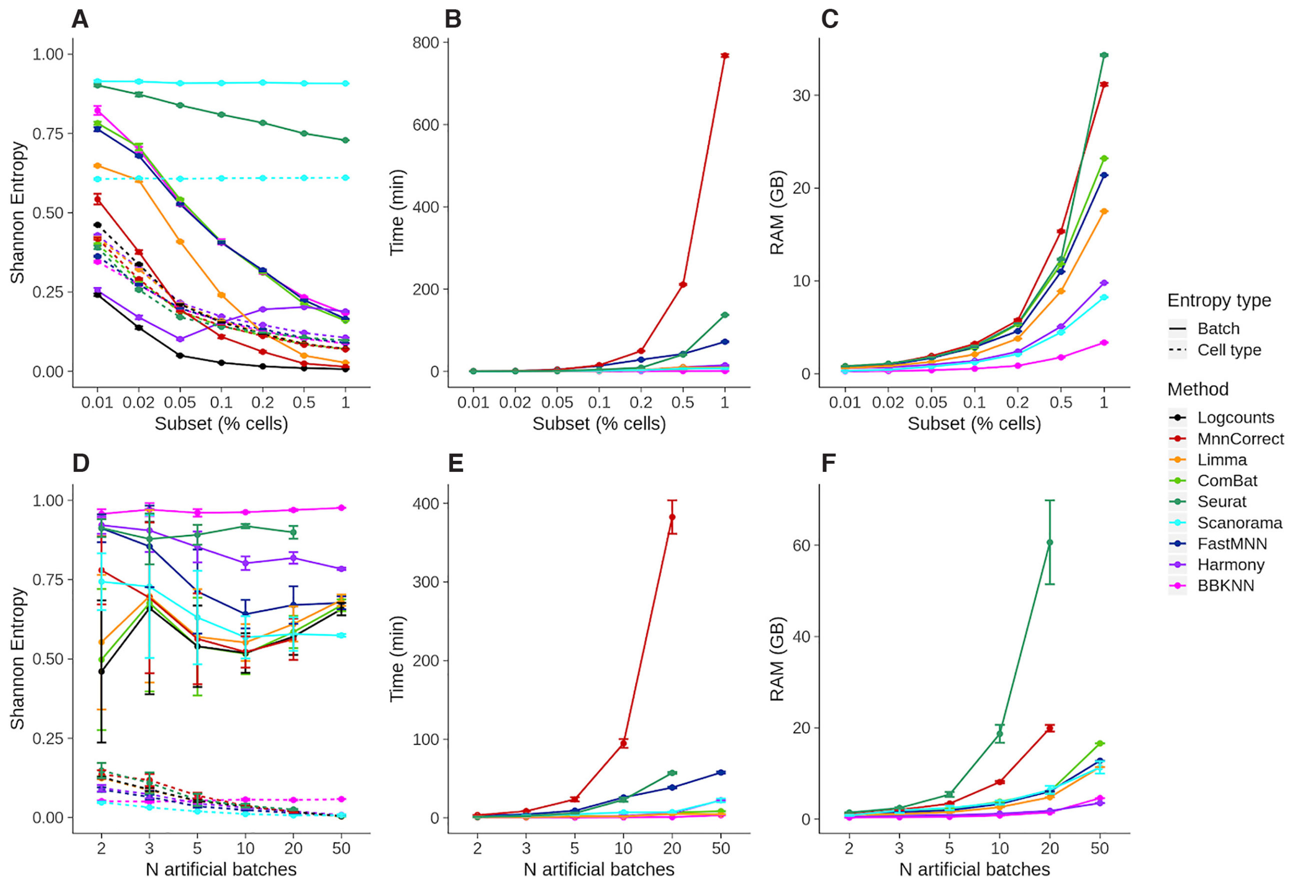

Figure 4. Entropy measures and resource consumption of methods as a function of the number of cells and simulated batches. (**A**) Batch and cell type entropies, (**B**) running time and (**C**) RAM usage over different subsets of the Tabula Muris atlas object with ∼61 000 cells in total. (**D**) Batch and cell type entropies, (**E**) running time and (**F**) RAM usage over an increasing number of simulated batches of 1001 cells each, generated from Tabula Muris atlas 10X cells.

We fixed the batch size to 1001 cells and we created datasets including 2, 3, 5, 10, 20 and 50 and batches, introducing small artificial batch effects. Cell type entropies are maintained low with the number of batches for all methods, highlighting the capacity of our batch simulating procedure to not mix distinct cell populations as batches are included. Regarding batch entropy (Figure 4D), BBKNN, Seurat and Harmony show the most stable performance as the number of batches increases. Although all methods have an exponential increase in both memory use and runtime, mnnCorrect stands out again as the slowest method. As before, we find that Seurat consumes the most memory, and along with mnnCorrect it fails to integrate 50 batches.

Impact of batch correction on unsupervised clustering and identification of marker genes

A key advantage of the entropy measures is that they can easily be calculated for any dataset containing discrete cell state clusters and that they are easy to interpret. However, they only evaluate the mixing of the cells as represented by the nearest neighbor graph, and they do not directly assess how the batch correction will impact downstream analyses based on the corrected data. To understand how specific aspects of the analysis are affected, tailored benchmarks are required. BatchBench allows users to add customized modules to evaluate the aspect they find most relevant. Here, we consider two common types of analyses, unsupervised clustering and identification of marker genes.

To evaluate the effect on unsupervised clustering, we apply four published methods, Leiden (33), Louvain (34), SC3 (35), RaceID (2) and hierarchical clustering, to the corrected data, and we then compare the merged cluster labels to the ones that were assigned prior to merging. To assess the proximity between clusterings we used a distance metric, variation of information, and a similarity metric, Adjusted Rand Index (ARI). The two measurements are by definition inversely correlated, and because they are consistent (Spearman's rho = –0.87) we will mainly refer to the ARI results. A common question regarding clustering refers to the choice of which features to include. To determine the effect of feature selection in clustering performance after batch correction we establish five fractions of features: 0.05, 0.1, 0.2, 0.5 and 1.0 (all of the features) by ranking genes descendingly by their coefficient of variation. Note that the feature selection could not be applied to BBKNN, Harmony and FastMNN since they do not operate in the gene space.

Our analysis of the MCA suggested small differences in cell type entropy, but large differences in how well the batches were mixed (Figure 3). By contrast, when running unsupervised clustering the batch correction methods achieve similar ARI values, except Race-ID kmeans and kmedoids, which perform worse. Closer inspection reveals large differences between tissues, something that is not evident from the entropy measures (Supplementary Table S1). In general a greater clustering similarity for this dataset is achieved by clustering with all genes (Supplementary Figure S10). Except for RaceID-kmeans, which in turn shows a very poor clustering similarity. Note RaceID clustering for Bone Marrow was interrupted after running for a week, and hence is not displayed.

For the Tabula Muris we observe a similar pattern with large differences in ARI between tissues and relatively small differences across methods. Compared to MCA, we observe an improvement in similarity values for RaceID, SC3 and hierarchical clustering algorithms, whereas Leiden and Louvain algorithms show worse performance. Closer inspection reveals that the Leiden and Louvain methods perform poorly for datasets with a small number of clusters (Supplementary Figures S3–S9). Surprisingly for heart and mammary glands, the best results are achieved by hierarchical clustering, RaceID and SC3 applied to the uncorrected data. There is a higher diversity in the feature fraction displaying the best similarity (Supplementary Figure S10). For TM datasets, the usage of a smaller fraction of features with a higher coefficient of variation results in an enhanced clustering.

For the pancreas dataset, hierarchical clustering together with RaceID and SC3 algorithms tend to have a higher ARI. Inclusion of all features in the clustering tend to yield better similarity results (Supplementary Figure S10). We also highlight that hierarchical clustering applied to BBKNN distance matrix is not a good approach. Additionally, Scanorama shows highly variable performance across the clustering algorithms and datasets considered.

The main objective of batch correction methods is to ensure that cells with similar expression profiles end up near each other. The most widely used metrics, e.g. mixing entropies or inverse Simpson index (16,19,29), are designed to evaluate this aspect. However, if a researcher is interested in analyzing the expression values for other purposes then it is important to make sure that the corrected values are close to the original ones. To investigate how much expression matrices are distorted by the different methods, we compared the marker genes identified before and after batch correction for the five methods that modify the expression matrix (Table 1). We identified marker genes for each batch individually as well as for the merged datasets from each method that outputs a modified expression matrix. Unlike the entropy and clustering analyses, we observed stark differences between batch correction methods. Remarkably, after merging using Scanorama or mnnCorrect, not a single marker gene is identified. Only ComBat and Limma are able to identify marker genes for most cell types, while Seurat only reports markers for a minority of cell types in most tissues (Figure 5B). Comparing the similarity between the marker genes identified in the individual batches and the merged dataset using a generalized Jaccard index (36), we find that Seurat provides the highest degree of consistency (Figure 5C). However, it is important to keep in mind that Seurat's good performance is biased by the fact that it reports marker genes for fewer cell types than the other methods. A similar problem stems from the fact that sometimes the individual batches do not share any or only few marker genes prior to merging, e.g. the neonatal calvaria from the MCA, which explains the grey boxes in Figure 5c.

DISCUSSION

We have developed BatchBench, a customizable pipeline for comparing scRNA-seq batch correction methods. We have assessed the performance of eight popular batch correction methods based on entropy measurements across three datasets, suffering from donor and platform effects. Our results highlight Seurat as the top performer as it correctly merges batches while maintaining the separation of distinct cell populations. Harmony also shows very good results in pancreas and MCA but surprisingly fails in correcting the Tabula Muris batch effects. Scanorama and fastMNN can be considered consistent good performers. Regarding BBKNN, we note that the entropies are not suitable for evaluating its performance as the method operates by identifying nearest neighbours in each of the provided batches (26) and adjusting neighbors to maximize the batch entropy. Hence, a different metric should be established to evaluate the performance of BBKNN. We also evaluated how the methods perform as the number of cells and the number of batches are varied. Here, we highlight Harmony as a method that provides good performance while being economical in its use of computational resources. However, our analyses suggest that all methods, with the possible exceptions of BBKNN and Harmony, will struggle to integrate hundreds of batches even if each batch is relatively small. Thus, improving scalability is a central requirement for future methods.

A key insight from our study is that the entropy measures do not fully reflect how the choice of batch correction method will impact downstream computational analyses. We applied five different unsupervised clustering methods to the merged datasets, and the results are not as clear as for the entropy analyses. No single method emerges as the best performer, and in some cases the best results were obtained using the uncorrected data. This result highlights the importance of using benchmarks that are more closely linked to the analysis that will be carried out for the merged dataset.

Our attempt to identify marker genes from the corrected dataset demonstrates the difficulty of using the merged expression matrix for downstream analyses. As none of the methods considered in our study performed adequately in this benchmark, we highlight this as an area where improvements are required. Since marker genes are not preserved, we stress the importance for users to monitor how expression levels change. Any analysis based on the expression levels, e.g. identification of marker genes or differentially expressed genes, will need to be verified to ensure that the result was not distorted due to the alterations introduced by the batch correction method. An important limitation of our marker gene analysis is that it only quantifies consistency as there is not yet an established ground truth for what marker genes are represented for the cell types in our study. We tried to use marker gene lists from the literature as

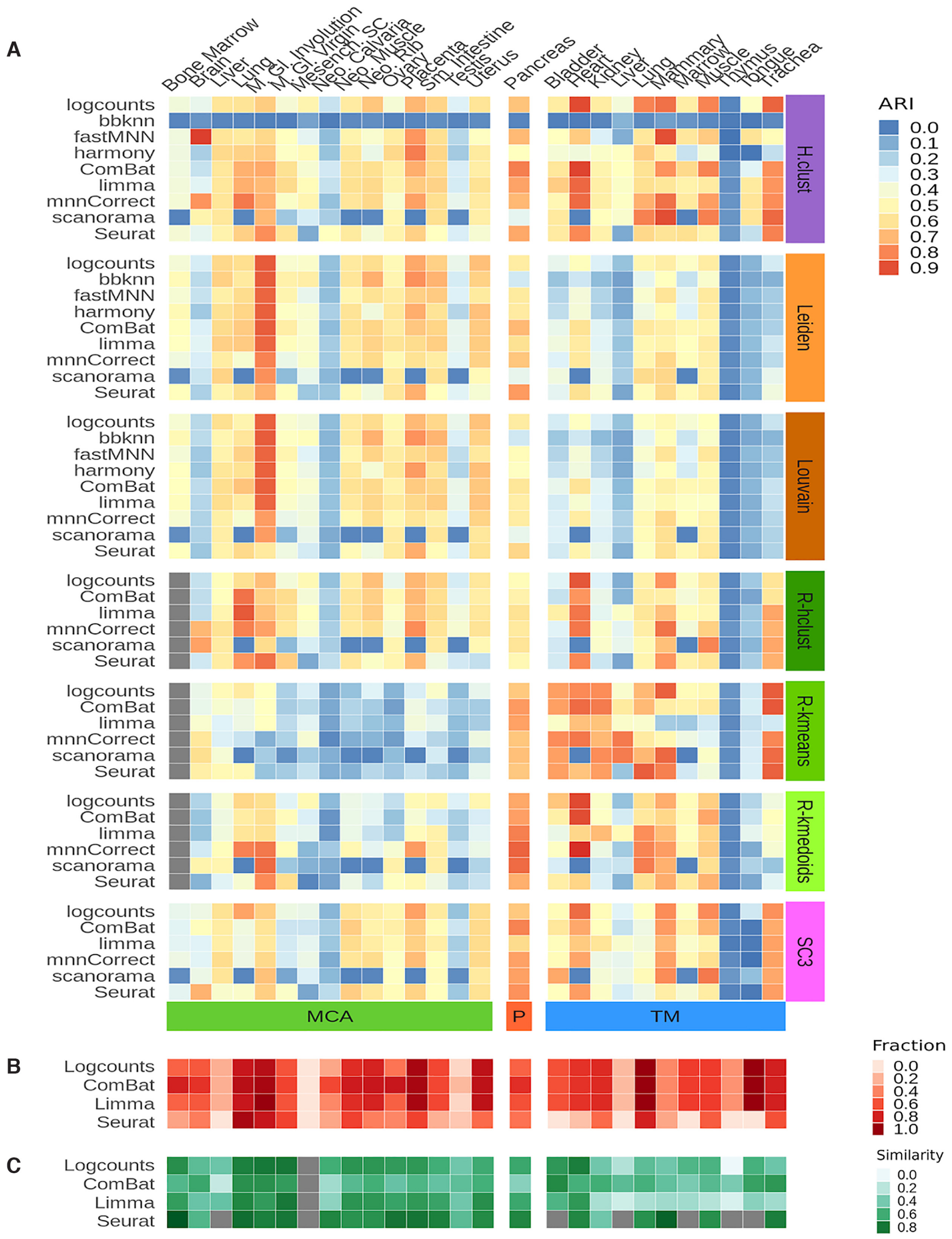

Figure 5. Evaluation of the impact of batch correction on unsupervised clustering and marker gene identification. (**A**) Clustering similarity of batch corrected output to cell labels as evaluated by the Adjusted Rand Index. The highest ARI value from the five fractions of features considered for the clustering is displayed. *MCA: Mouse Cell Atlas, P: Pancreas, TM: Tabula Muris*. (**B**) Fraction of total cell types over which marker genes are detected. (**C**) Similarity of marker genes between merged dataset and individual batches as evaluated by the generalized Jaccard Index.

represented by the CellMarker database (37), but we found that all pancrease datasets provided poor overlap, even before batch correction (Supplementary Figure S15).

Benchmark studies are important as they help guide researchers in their choice of methods. They are also helpful for developers as they can highlight limitations of existing methods and provide guidance as to where improvements are needed. One shortcoming of traditional benchmarks, however, is that they are static in nature and that they only consider the datasets that the authors of the benchmark study had chosen to include. A related issue is that the metrics used to evaluate methods may not be relevant to all datasets and research questions. Along with a similar study by Leucken *et al*. (40), BatchBench will serve as a useful platform to the community as it enables benchmarks to be tailored to specific needs.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We would like to thank members of the Cellular Genetics Informatics team and the Hemberg lab for constructive feedback and comments. The S.vD., V.Y.K. and M.H. were funded by a core grant from the Wellcome Trust. R.C.G. was funded by the Polytechnic University of Valencia under an Erasmus+ studentship, and by the Wellcome Trust.
Author contributions: M.H. and V.Y.K. conceived of the study and supervised the work. R.C.G. and S.vD. developed the Nextflow pipeline. R.C.G. carried out the benchmarking of the three datasets and eight methods. R.C.G. and M.H. wrote the manuscript with inputs from V.Y.K. and S.vD.

FUNDING

Wellcome Trust; Polytechnic University of Valencia [Erasmus+]. Funding for open access charge: Wellcome Trust.
Conflict of interest statement. None declared.

REFERENCES

1. Rostom,R., Svensson,V., Teichmann,S.A. and Kar,G. (2017) Computational approaches for interpreting scRNA-seq data. *FEBS Lett.*, **591**, 2213–2225.
2. Grün,D., Lyubimova,A., Kester,L., Wiebrands,K., Basak,O., Sasaki,N., Clevers,H. and van Oudenaarden,A. (2015) Single-cell messenger RNA sequencing reveals rare intestinal cell types. *Nature*, **525**, 251–255.
3. Zeisel,A., Muñoz-Manchado,A.B., Codeluppi,S., Lönnerberg,P., La Manno,G., Juréus,A., Marques,S., Munguba,H., He,L., Betsholtz,C. *et al.* (2015) Brain structure. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, **347**, 1138–1142.
4. Bendall,S.C., Davis,K.L., Amir,E.-A.D., Tadmor,M.D., Simonds,E.F., Chen,T.J., Shenfeld,D.K., Nolan,G.P. and Pe'er,D. (2014) Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development. *Cell*, **157**, 714–725.
5. Haghverdi,L., Büttner,M., Wolf,F.A., Buettner,F. and Theis,F.J. (2016) Diffusion pseudotime robustly reconstructs lineage branching. *Nat. Methods*, **13**, 845–848.
6. Lönnberg,T., Svensson,V., James,K.R., Fernandez-Ruiz,D., Sebina,I., Montandon,R., Soon,M.S.F., Fogg,L.G., Nair,A.S., Liligeto,U. *et al.* (2017) Single-cell RNA-seq and computational analysis using temporal mixture modelling resolves Th1/Tfh fate bifurcation in malaria. *Sci Immunol.*, **2**, eaal2.
7. Shalek,A.K., Satija,R., Adiconis,X., Gertner,R.S., Gaublomme,J.T., Raychowdhury,R., Schwartz,S., Yosef,N., Malboeuf,C., Lu,D. *et al.* (2013) Single-cell transcriptomics reveals bimodality in expression and splicing in immune cells. *Nature*, **498**, 236–240.
8. Marinov,G.K., Williams,B.A., McCue,K., Schroth,G.P., Gertz,J., Myers,R.M. and Wold,B.J. (2014) From single-cell to cell-pool transcriptomes: stochasticity in gene expression and RNA splicing. *Genome Res.*, **24**, 496–510.
9. Qiu,X., Hill,A., Packer,J., Lin,D., Ma,Y.-A. and Trapnell,C. (2017) Single-cell mRNA quantification and differential analysis with Census. *Nat. Methods*, **14**, 309–315.
10. Welch,J.D., Hu,Y. and Prins,J.F. (2016) Robust detection of alternative splicing in a population of single cells. *Nucleic Acids Res.*, **44**, e73.
11. Trapnell,C., Hendrickson,D.G., Sauvageau,M., Goff,L., Rinn,J.L. and Pachter,L. (2013) Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nat. Biotechnol.*, **31**, 46–53.
12. Deng,Q., Ramsköld,D., Reinius,B. and Sandberg,R. (2014) Single-cell RNA-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science*, **343**, 193–196.
13. Kim,J.K., Kolodziejczyk,A.A., Ilicic,T., Teichmann,S.A. and Marioni,J.C. (2015) Characterizing noise structure in single-cell RNA-seq distinguishes genuine from technical stochastic allelic expression. *Nat. Commun.*, **6**:8687.
14. Reinius,B., Mold,J.E., Ramsköld,D., Deng,Q., Johnsson,P., Michaëlsson,J., Frisén,J. and Sandberg,R. (2016) Analysis of allelic expression patterns in clonal somatic cells by single-cell RNA-seq. *Nat. Genet.*, **48**, 1430–1435.
15. Kim,J.K. and Marioni,J.C. (2013) Inferring the kinetics of stochastic gene expression from single-cell RNA-sequencing data. *Genome Biol.*, **14**, R7.
16. Haghverdi,L., Lun,A.T.L., Morgan,M.D. and Marioni,J.C. (2018) Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat. Biotechnol.*, **36**, 421–427.
17. Hwang,B., Lee,J.H. and Bang,D. (2018) Single-cell RNA sequencing technologies and bioinformatics pipelines. *Exp. Mol. Med.*, **50**, 96.
18. Chen,G., Ning,B. and Shi,T. (2019) Single-Cell RNA-Seq technologies and related computational data analysis. *Front. Genet.*, **10**, 317.
19. Tran,H.T.N., Ang,K.S., Chevrier,M., Zhang,X., Lee,N.Y.S., Goh,M. and Chen,J. (2020) A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biol.*, **21**, 12.
20. Di Tommaso,P., Chatzou,M., Floden,E.W., Barja,P.P., Palumbo,E. and Notredame,C. (2017) Nextflow enables reproducible computational workflows. *Nat. Biotechnol.*, **35**, 316–319.
21. Ritchie,M.E., Phipson,B., Wu,D., Hu,Y., Law,C.W., Shi,W. and Smyth,G.K. (2015) limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res.*, **43**, e47.
22. Leek,J.T., Johnson,W.E., Parker,H.S., Jaffe,A.E. and Storey,J.D. (2012) The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics*, **28**, 882–883.
23. Stuart,T., Butler,A., Hoffman,P., Hafemeister,C., Papalexi,E., Mauck,W.M., Hao,Y., Stoeckius,M., Smibert,P. and Satija,R. (2019) Comprehensive integration of single-cell data. *Cell*, **177**, 1888–1902.
24. Hie,B., Bryson,B. and Berger,B. (2019) Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. *Nat. Biotechnol.*, **37**, 685–691.
25. Korsunsky,I., Millard,N., Fan,J., Slowikowski,K., Zhang,F., Wei,K., Baglaenko,Y., Brenner,M., Loh,P.R. and Raychaudhuri,S. (2019) Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat. Methods*, **16**, 1289–1296.
26. Polański,K., Young,M.D., Miao,Z., Meyer,K.B., Teichmann,S.A. and Park,J.-E. (2020) BBKNN: fast batch alignment of single cell transcriptomes. *Bioinformatics*, **36**, 964–965.
27. Muraro,M.J., Dharmadhikari,G., Grün,D., Groen,N., Dielen,T., Jansen,E., van Gurp,L., Engelse,M.A., Carlotti,F., de Koning,E.J. *et al.* (2016) A single-cell transcriptome atlas of the human pancreas. *Cell Syst.*, **3**, 385–394.
28. Segerstolpe,Å., Palasantza,A., Eliasson,P., Andersson,E.-M., Andréasson,A.-C., Sun,X., Picelli,S., Sabirsh,A., Clausen,M.,

Bjursell,M.K. *et al.* (2016) Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes. *Cell Metab.*, **24**, 593–607.

29. Azizi,E., Carr,A.J., Plitas,G., Cornish,A.E., Konopacki,C., Prabhakaran,S., Nainys,J., Wu,K., Kiseliovas,V., Setty,M. *et al.* (2018) Single-cell map of diverse immune phenotypes in the breast tumor microenvironment. *Cell*, **174**, 1293–1308.
30. Kiselev,V.Y., Yiu,A. and Hemberg,M. (2018) scmap: projection of single-cell RNA-seq data across data sets. *Nat. Methods*, **15**, 359–362.
31. Han,X., Wang,R., Zhou,Y., Fei,L., Sun,H., Lai,S., Saadatpour,A., Zhou,Z., Chen,H., Ye,F. *et al.* (2018) Mapping the mouse cell atlas by microwell-seq. *Cell*, **172**, 1091–1107.
32. Tabula Muris Consortium (2018) Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature*, **562**, 367–372.
33. Traag,V.A., Waltman,L. and van Eck,N.J. (2019) From Louvain to Leiden: guaranteeing well-connected communities. *Sci. Rep.*, **9**, 5233.
34. Blondel,V.D., Guillaume,J.-.L., Lambiotte,R. and Lefebvre,E. (2008) Fast unfolding of communities in large networks. *J Stat Mech*, **2008**, P10008.
35. Kiselev,V.Y., Kirschner,K., Schaub,M.T., Andrews,T., Yiu,A., Chandra,T., Natarajan,K.N., Reik,W., Barahona,M., Green,A.R. *et al.* (2017) SC3: consensus clustering of single-cell RNA-seq data. *Nat. Methods*, **14**, 483–486.
36. Diserud,O.H. and Odegaard,F. (2007) A multiple-site similarity measure. *Biol. Lett.*, **3**, 20–22.
37. Zhang,X., Lan,Y., Xu,J., Quan,F., Zhao,E., Deng,C., Luo,T., Xu,L., Liao,G., Yan,M. *et al.* (2019) CellMarker: a manually curated resource of cell markers in human and mouse. *Nucleic Acids Res.*, **47**, D721–D728.
38. McInnes,L., Healy,J., Saul,N. and Großberger,L. (2018) UMAP: uniform manifold approximation and projection. *JOSS*, **3**, 861.
39. Baron,M., Veres,A., Wolock,S.L., Faust,A.L., Gaujoux,R., Vetere,A., Ryu,J.H., Wagner,B.K., Shen-Orr,S.S., Klein,A.M. *et al.* (2016) A single-cell transcriptomic map of the human and mouse pancreas reveals inter- and intra-cell population structure. *Cell Syst.*, **3**, 346–360.
40. Luecken,M.D. and Theis,F.J. (2019) Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol. Syst. Biol.*, **15**, e8746.