Unbiased integration of single cell transcriptomes using a linear hybrid method

Martin Loza Lopez[1], Shunsuke Teraguchi[1,2], Diego Diez[1,*]

[1]Immunology Frontier Research Center, Osaka University, 3-1 Yamada-oka, Suita, Osaka 565-0871, Japan

[2]Faculty of Data Science, Shiga University, Hikone, 522-8522, Japan

* Corresponding author: diez@ifrec.osaka-u.ac.jp

Abstract

Single cell transcriptomic approaches are becoming mainstream, with replicated experiments performed in the same single cell platform being more common. Methods that enable to integrate these datasets by removing batch effects while preserving biological information are required. Here we present Canek, a method that, leveraging information from mutual nearest neighbors, combines a local linear correction with a cell-specific non-linear correction using fuzzy logic. Using a combination of real and synthetic datasets we show that Canek corrects batch effects while introducing the least amount of bias when compared with competing methods. Canek is computationally efficient, being able to integrate single cell transcriptomes for thousands of cells from replicated experiments.

Introduction

Single-cell transcriptomic approaches are revolutionizing many fields in biology by enabling unbiased interrogation of transcriptional states at the single-cell level. The number of published single-cell genomics datasets is increasing rapidly [1]. Integration and reanalysis of these datasets have the potential to provide insight into biological processes by combining cells and conditions from different studies. However, this process is hindered by the existence of batch effects [2, 3].

Many methods exist that aim to integrate datasets obtained from the same tissues but using different technologies [4]. One of the pioneering techniques is the so-called Mutual Nearest Neighbors (MNN) correction method available in the R package batchelor [5]. In this method, MNN pairs are used to identify corresponding cells from the two batches. A pair-specific correction vector is calculated as the difference between expression profiles of the cells from each MNN pair. The correction vectors are then weighted using a Gaussian kernel to smooth the corrections between adjacent cells. A critical assumption is that at least one cell is shared among batches and that the batch effect is almost orthogonal to the biological space (see [5] for a discussion of these assumptions). Other tools have used the idea of using MNNs to integrate batches [6-8]. Another popular method is the one implemented in the Seurat R package, which finds MNN pairs in a correlated space using canonical correlation analysis (CCA) [6]. The identified pairs are used as “anchors” to correct batch effects on the input batches. Another interesting approach is the one available in Harmony, in which the batch effects are iteratively removed by clustering in a low dimensional space among input batches [9]. LIGER applies a similar clustering approach by segmenting cells using a shared factor neighborhood graph under

a low dimensional space defined with an integrative non-negative matrix factorization method [10].

Recently, a very comprehensive benchmarking of 14 batch correction methods including the ones mentioned above was published [4]. The methods were tested under different scenarios including the integration of different technologies, non-similar cells, large datasets, multiple batches, and simulated data. The top three scored methods were Harmony, Liger, and Seurat. However, they found that each method performed differently on each test with no superior method [4].

An open question is whether batch correction methods introduce biases in the data that disturb the biological information and alter the structure of the cell populations. As single-cell genomics technologies become mainstream, more laboratories will perform experiments consisting of different experimental conditions with biological replicates from the same technology. In this setting, integration of datasets with minimal impact to cell type identities is essential. Here, we present a new method called Canek that enables efficient integration of replicated experiments with minimum bias. Canek leverages information from MNNs and combines a hybrid linear/non-linear framework to identify and correct cell type specific batch effects. Using a combination of real data and simulations we show that, unlike current batch correction/integration methods, Canek enables unbiased correction of single cell transcriptome data.

Results

Canek successfully corrects batch effects in the Jurkat/293T dataset

We introduce a new method called Canek that enables efficient integration while preserving data structure (see Methods for a detailed description). We assume a mostly linear batch effect with small non-linearities between a pair of datasets to be corrected, which we treat as the *reference batch* and the *query batch* (Figure 1a). We define *batch effect observations* using *mutual nearest neighbors* (MNN) pairs [5] and distinguish cell groups from the *query batch* using *clustering* (Figure 1b). We estimate a *correction vector* for each cluster using the median gene expression differences between the cells in each membership of the query batch and the corresponding cells of the reference identified by the MNN pairs (arrows on Figure 1c). The correction vector can thus be used to remove the batch effect from each membership in the query batch. In this linear correction, the same correction is applied to all the cells from the same membership (Figure 1c). Then, we perform a *non-linear correction* by calculating a cell-specific correction using fuzzy logic. This is done by defining a minimum spanning tree among memberships and then smoothing the transitions between the correction vectors (Figure 1d).

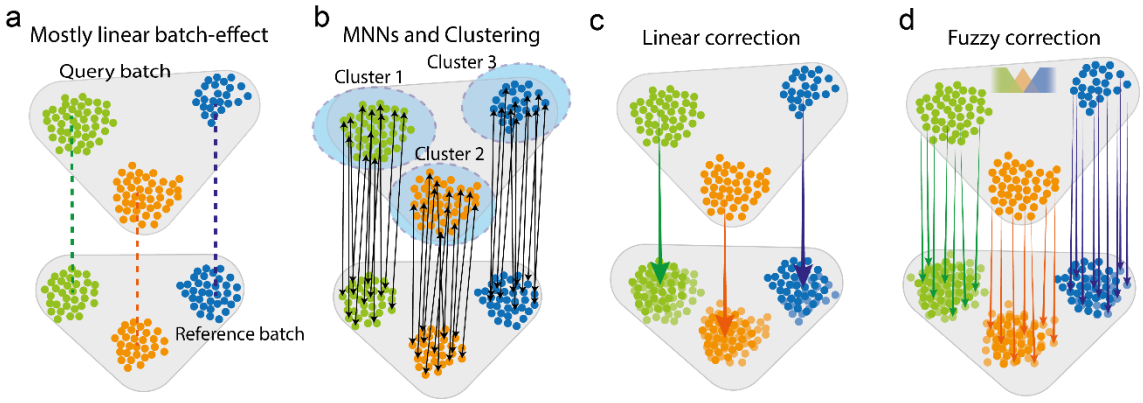

Figure 1 ***Overview of Canek workflow****.* ***a)*** We start with a *reference batch and query batch, assuming a predominantly linear batch effect.* ***b)*** *Cell clusters are defined on the query batch and MNN pairs (arrows) are used to define batch effect observations.* ***c)*** *The MNN pairs from each cluster are used to estimate membership specific correction vectors. These vectors can be used to correct the batch effect or,* ***d)*** *a non-linear correction is implemented by calculating cell-specific correction vectors using fuzzy logic.*

To illustrate the ability of Canek to correct batch effects we use a dataset consisting of three batches, one made of 293T HEK cells, one of Jurkat cells, and the last one being a 50:50 mixture of 293T and Jurkat cells [11]. Principal component analysis (PCA) of the uncorrected data shows that although 293T cells align almost perfectly, there is a considerable separation in the Jurkat cell population (Figure 2a PCA). This bias is also clear from the Uniform Manifold Approximation and Projection map (UMAP) [12] (Figure 2a UMAP). Looking at cell specific markers we can see that the aligned cells express XIST whereas the unaligned cells express CD3D, indicating they are 293T cells and Jurkat cells respectively. We consider this bias to be due to batch effect. We applied batch correction with Seurat, MNN, scMerge and Combat [5, 6, 13, 14]. MNN corrected the batch effect and enabled the identification of the expected cell population clusters (Figure 2b). However, other methods like the ones implemented in the Seurat package and scMerge, resulted in mixed cell populations (Figure 2c,d). Even a linear method like Combat incorrectly mixed cell types (Figure 2e). Canek was able to successfully correct the batch effect in this scenario (Figure 2f).

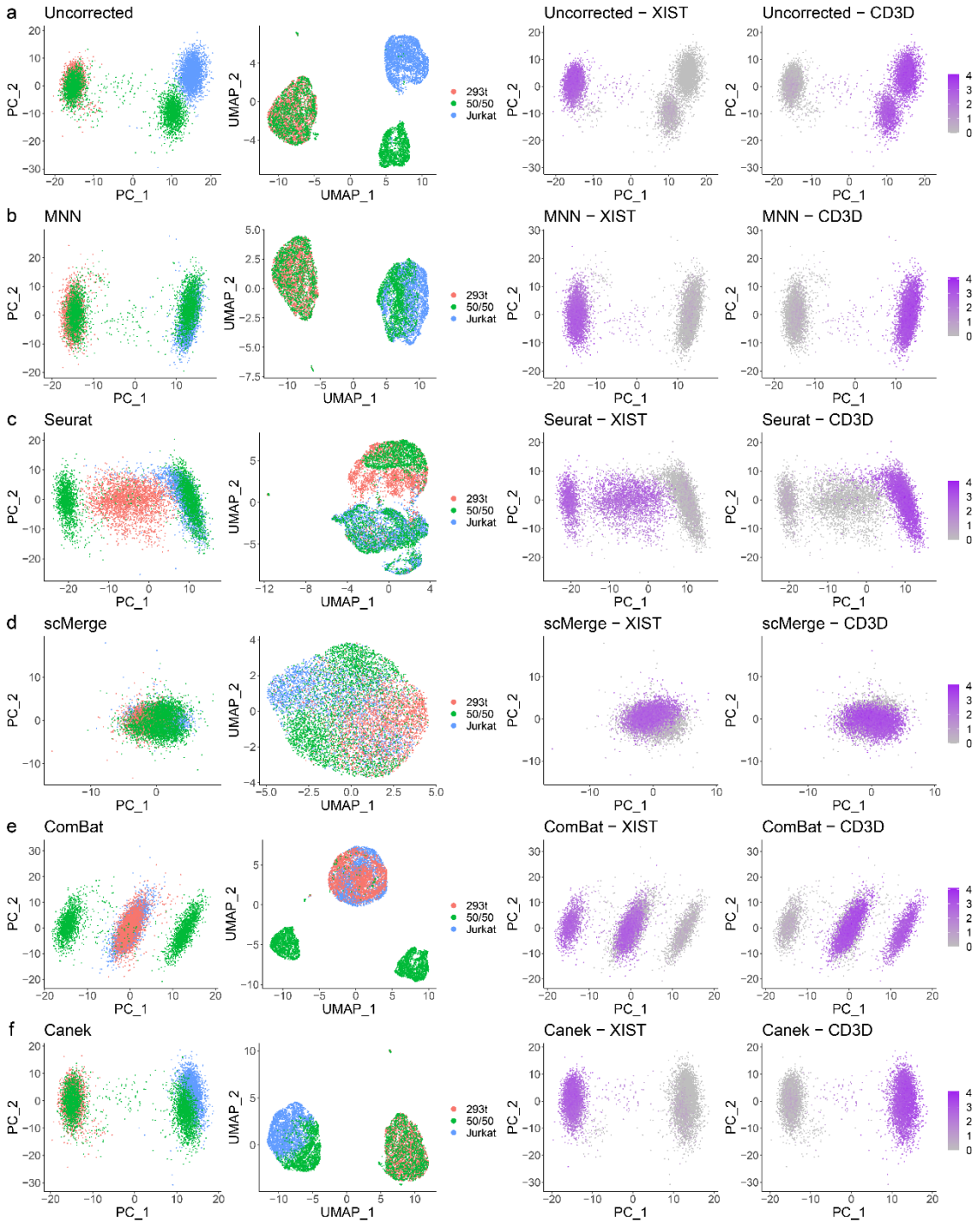

Figure 2 ***Batch effect correction methods may incorrectly mix dissimilar cell types.*** *Batch effect correction of three batches, two containing pure Jurkat and HEK293T cells, and one with a 50:50 mix of Jurkat and HEK293T cells. Jurkat and HEK293T cells are characterized by the expression of CD3D and XIST genes respectively.* ***a)*** *Before correction Jurkat cells grouped by batch.* ***b-f)*** shows the results of batch *effect correction using MNN, Seurat, scMerge, ComBat and Canek. MNN and Canek correctly integrated the Jurkat cells. Other methods like Seurat, scMerge and ComBat incorrectly mixed Jurkat and HEK293T cells.*

Evaluation of correction bias

One goal of Canek is to integrate datasets coming from replicated experiments while preserving as much as possible the original biological structure. We define batch correction bias as undesired correction that may alter the original biological structure. To quantify how much bias batch correction methods introduce, we use a pseudo-batch approach. Starting from a single dataset we identify clusters and generate pseudo-batches by sampling cells without replacement. We argue that batch correction methods should not correct in this scenario since no batch effect exists, and identification of clusters from the integrated batches should preserve as much as possible the clusters from the uncorrected dataset. To tests this hypothesis, we generated pseudo-batches from public datasets and applied batch correction using several methods. To measure how much bias was introduced we computed kBET and silhouette scores for the uncorrected dataset and after correction. Since there is no batch effect, the metric scores for the uncorrected dataset corresponds to the best possible values to be obtained after correction.

Figure 3a shows the results from applying this strategy to the spleen data from Tabula Muris [15]. The first plot shows the cell clusters with colors and the next two show the two pseudo-batches. In Figures 3b-i we show the pseudo-batches after integration with several methods. Except for MNN, that clearly shows a failure to integrate the pseudo batches, it is difficult to judge from these plots how well the different methods performed. Therefore, we show the performance of each method using two metrics: kBET and silhouette scores. Figure 3j shows the scatterplot of the silhouette and kBET scores obtained in this experiment. In this plot, the dashed lines indicate the values for the uncorrected dataset. The crossing point represents the values obtained when no bias exists and so is the optimal value. This shows that ComBat, Harmony, and Canek are the methods that lead to scores closer to the target. To estimate the variability of the results due to pseudo-batch sampling, we repeated each experiment 10 times. In Supplementary Figure 1a, we show silhouette vs. kBET scores for the 10 experiments together with the average of each kBET/silhouette replicates, as well as an ellipse indicating the dispersion. Supplementary Figure 1b is a zoom into the target values and shows that Canek is the method that obtained scores closest to the values of the uncorrected dataset. This demonstrates that Canek introduces the least amount of bias when no batch effect is present.

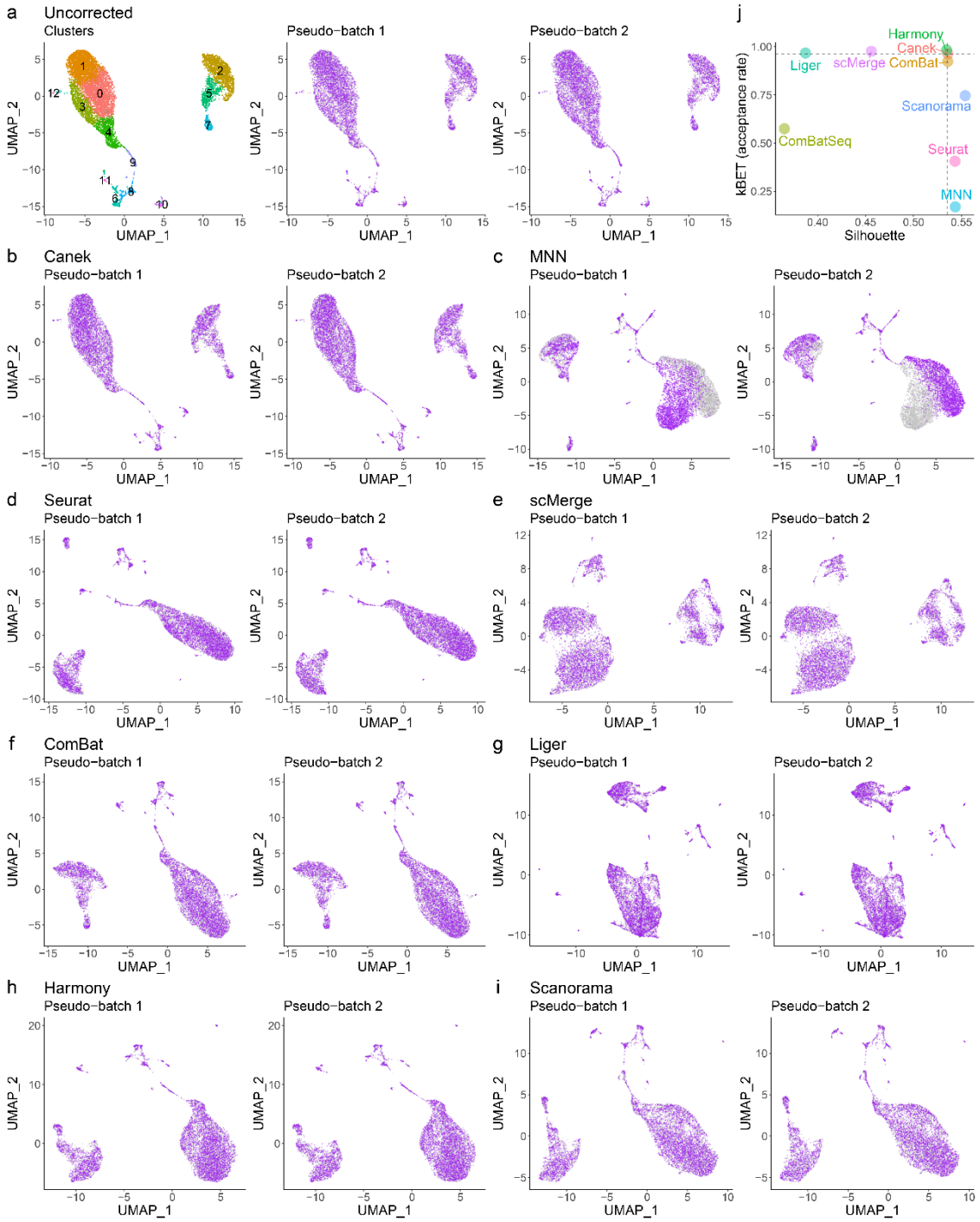

Figure 3 ***Correction methods may introduce biases when there is no batch effect****. Analysis of pseudo-batches using the Tabula Muris spleen dataset. The uncorrected batch serves as the gold standard to compare the integrations. An unbiased integration would retain the cell mixing and clustering from the gold standard.* ***a)*** *The uncorrected data is clustered and pseudo-batches are created by sampling.* ***b-i)*** *The pseudo-batches are integrated with different batch effect correction methods.* ***j)*** *Using kBET and silhouette metrics, the mixing among batches and the cluster preservation are evaluated. The scores from the gold standard are shown as gray dashed lines, while the scores from the correction methods are indicated as colored points. Unbiased methods are those whose metrics are closest to the intersection of the gray lines.*

Evaluation of integration in simulated data

To estimate the ability to correct batch effects when the batch effect is known, we compared Canek with other methods using simulated data. We simulated three batches with some shared cell types using the splatter package [16]. One advantage of using splatter is that it allows us to remove the batch effect and obtain an integrated dataset to use as a gold standard (GS). Batch 1 is composed of two shared and one unique cell type, whereas batch 2 and 3 have one shared and one unique cell types (see Table 2 for a complete description). Figure 4a shows the PCA and the UMAP from the GS, Figure 4b the uncorrected datasets, and Figures 4c-f the results from four correction methods. The colors represent the batches and the cell type composition. We calculated the kBET and silhouette scores from the GS, the uncorrected data, and the integrated datasets. We expect the best correction methods to be close to the metrics from the GS. Supplementary Figure 2 shows the silhouette/kBET scatterplot where the dashed lines represent the scores from the GS, and the points represent the scores from the uncorrected and the corrected datasets from eight correction methods. Canek is the method closest to the scores of the GS. This is consistent with the PCA and UMAP plots in Figure 4c, where Canek has corrected the differences from the shared cell types. Interestingly, Harmony returned scores very close to the uncorrected data, suggesting that performed almost no correction (Supplementary Figure 2).

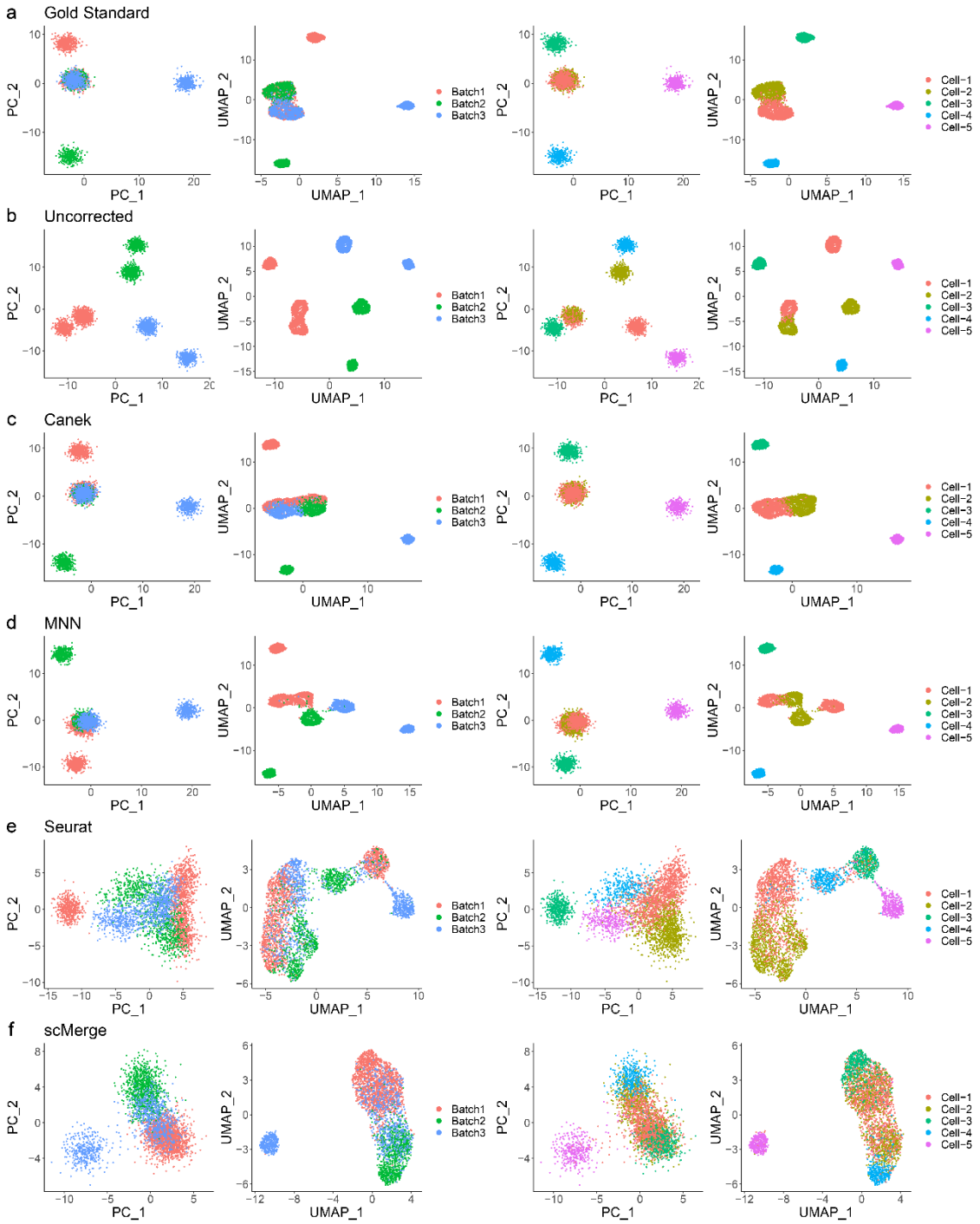

Figure 4 ***Batch effect correction on simulated data with a known gold standard.*** *Three batches were simulated to test the integration methods in a scenario with a known gold standard* ***a)****. Only two cell types, Cell-1 and Cell-2, are shared among batches.* ***b)*** *Uncorrected datasets.* ***c-f)*** *Batch effects correction with four methods. Canek* ***c)*** *stands as the best method in this scenario correctly integrating the shared cell types and maintaining the non-shared ones.* ***d)*** *MNN correction failed to integrate cells from the same type.* ***e,f)*** *Seurat and scMerge incorrectly mixed non-shared cell types.*

Application to real datasets

Next, we compared Canek with other methods on 3 real datasets: Tabula Muris spleen, human pancreatic islets, and interferon beta stimulation [15, 17-22]. We used a standardized pipeline in which the top variable features selected by Seurat for the integration were used with all methods.

First, we wanted to test the scenario in which the same sample is used with two different technologies simultaneously. Therefore, we integrated the Drop-seq and FACS batches from the Tabula Muris spleen datasets [15]. Supplementary Figure 3 shows the uncorrected data, and the correction done by Canek and four other correction methods. Except for Combat-seq, all the methods shown successfully integrated the datasets with cells from the same type found in the same clusters. This suggests Canek can integrate datasets from different technologies.

Next, we wanted to test the scenario in which similar tissues are used with different technologies. For this we integrated eight human pancreatic islets datasets from five different technologies. Figure 5a shows the uncorrected data, where the batch effect caused the cells to cluster by batch. Figure 5b-f shows the result of different integration methods. We can observe that methods like MNN and Seurat mix the datasets almost perfectly. Canek is able to integrate the batches but some differences between datasets are shown. We notice that these differences are correlated with disease state (Figure 6), with some of the samples containing type 2 diabetes whereas other containing only healthy individuals. Therefore, the observed differences can be due to biological differences.

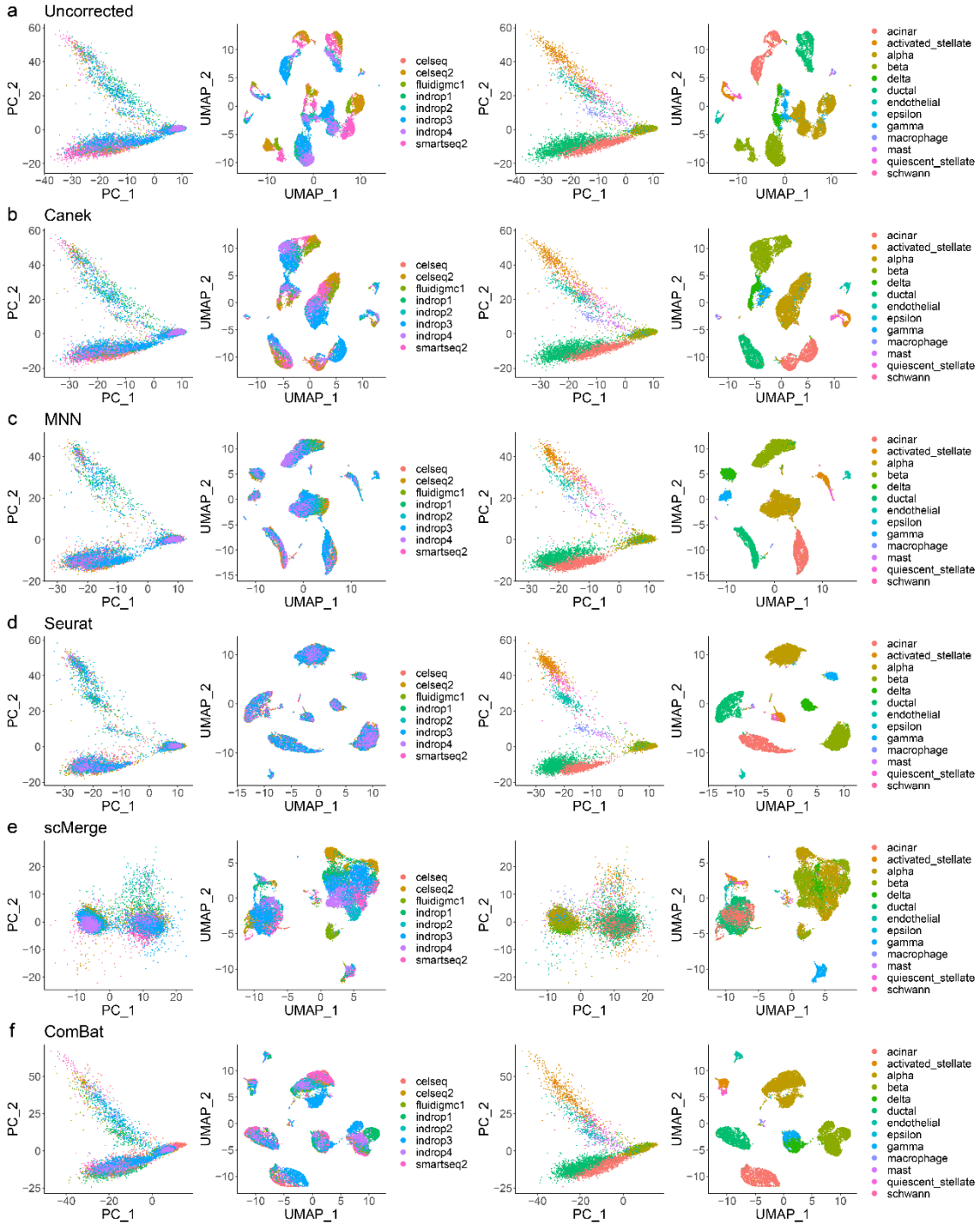

Figure 5 ***Integration datasets from different technologies.*** *Eight pancreatic datasets obtained using different technologies were corrected.* ***a)*** *Batch effects caused the cells to cluster by batch instead of by cell type.* ***c-f)*** *The batches were integrated using different methods.*

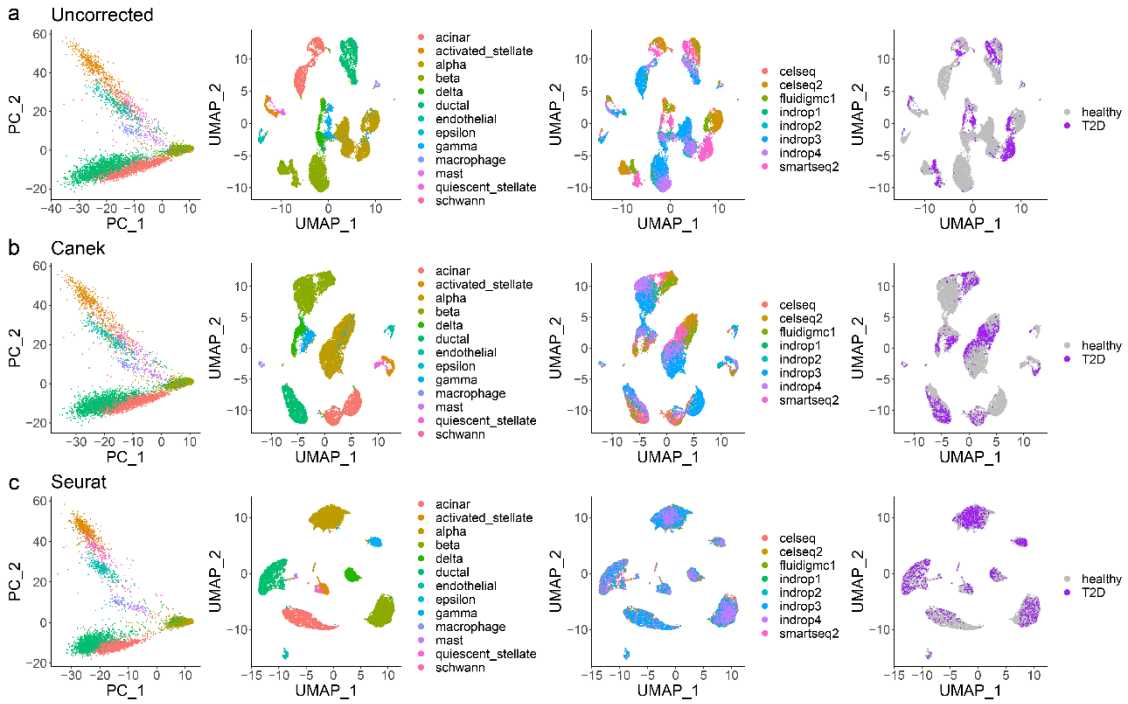

Figure 6 ***Latent conditions in the pancreatic integration.*** *Differences in the integration of some batches in Canek are correlated with disease state, suggesting that differences in batch integration are due to biological differences.*

Finally, we wanted to evaluate the scenario in which two samples are assayed using the same technology. For this we integrated a dataset obtained from PBMCs with and without interferon-beta stimulation [22]. In this scenario, differences between cell types are expected. Supplementary Figure 4a shows that in the uncorrected data, cells separate by batch. Supplementary Figures 4b-f shows the correction done by Canek and four other methods. After integrating with Canek, B cells and T cells are almost completely integrated but some differences remain. Differences in monocytes and dendritic cells in the stimulated vs. non-stimulated cells are more prominent. This is in agreement with experiments showing that interferon beta induces stronger changes in gene expression in monocytes compared to T cells [23].

Computational performance

To compare the computational performance and scalability of the different integration methods, we simulated datasets using splatter and recorded the integration time of Canek and other eight batch correction methods. We fixed the number of genes to 2,000 and varied the number of cells in the range of 10k to 100k. Figure 7 shows running time as a function of the number of cells. The fastest method was Combat, followed by Scanorama, Harmony, and Canek, all of them showing near linear increase in running time and able to integrate 100k cells in under 20 min. On the other side of the spectrum MNN, Seurat, and CombatSeq showed a non-linear increase in running time. These results demonstrate that Canek is a scalable method that can integrate hundreds of thousands of single-cell transcriptomes efficiently.

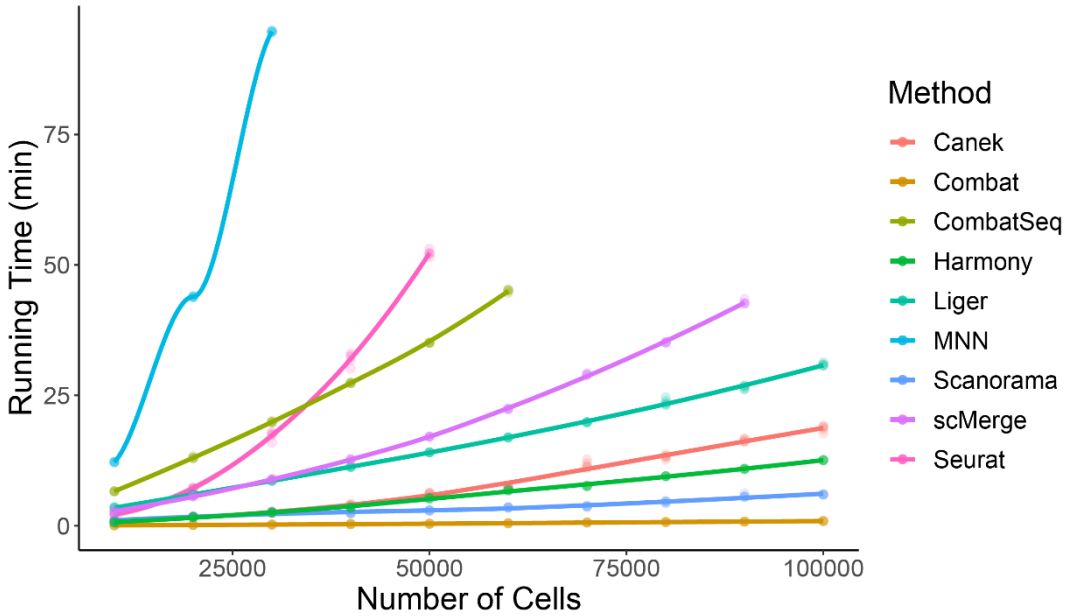

Figure 7 ***Runtime benchmark of Canek and other eight batch correction methods.*** *Each method was run 5 times on different datasets with the number of genes fixed to 2k and the number of cells varying in a range of 5k to 100k. The color code differentiates each of the methods, the dots represent the runtime, and the lines represent the time increasing trends. Canek displayed a linear time increase over these conditions.*

Discussion

Existing batch effect correction methods focus on integrating single-cell transcriptomics datasets obtained from different technologies and/or species, minimizing the differences among batches to obtain correlated cell types. While these frameworks offer a powerful solution to integrate datasets with strong differences between batches, they may introduce significant biases to favor a global integration. This could be a potential problem when these methods are applied to, e.g. replicated experiments obtained with the same technology, where we would like to preserve biological differences for downstream analyses such as clustering and differential gene expression. Canek provides an unbiased batch effect correction method for single-cell transcriptomics data that is suited for the integration of replicated experiments. We focused on preserving the inherent biological structure while being flexible enough to deal with small non-linear differences that might appear on heterogeneous datasets. We applied Canek on simulated and real datasets showing its ability to correct batch effects without masking local differences. We also tested Canek on a pseudo-batch scenario with no batch effect, being the method that best preserved the biological structure and introduced the least amount of bias.

We showed that Canek can successfully integrate datasets from different technologies (e.g., the Tabula Muris spleen dataset). Depending on the nature of the dataset Canek does not necessarily lead to the best batch mixing (e.g., in the human pancreatic islet integration). However, a more detailed analysis shows that latent variables other than batch may be influencing the mixing of datasets (e.g., including cells from healthy and disease donors, or unstimulated and stimulated cells).

We conservatively chose Canek's default parameters to work under diverse scenarios, but these might not be always optimal. For instance, a drawback of correction methods that use MNNs

pairs to define batch observations (i.e., MNN, Seurat, and Canek) is the assumption of having at least a shared cell type among batches. When this assumption is violated or the number of shared cells is low, it might be difficult to correctly integrate the datasets.

As single-cell RNA-seq from replicated experiments using the same technology become more common, batch effect correction methods that conserve local differences will be required. Canek provides a solution to this problem with an unbiased and computationally efficient batch effect correction.

Methods

Canek workflow

Figure 2 shows the workflow for correcting a pair of batches. We define one of the batches as the *query batch* and the other one as the *reference batch*. We correct the cells from the query batch to match the cells from the reference batch. When correcting more than two batches we perform an optional hierarchical optimization of batch order (see *Hierarchical integration* section). The main steps of Canek's workflow are:

1. Obtaining **batch effect observations** using mutual nearest neighbors (MNN) pairs.
2. **Clustering** the query batch to define local memberships.
3. Calculating a batch effect **correction vector** for each membership.
4. Obtaining a **fuzzy correction** by smoothing the transitions between the local correction vectors.

We expect the input datasets to be log normalized. The output dataset retains the same dimensionality (number of genes) as the input batches.

Batch effect Observations

The first step is to identify what we call batch effect observations. This is the gene expression differences of a set of cells from the reference and query batches that will enable us to estimate the batch effect.

To speed up computation we calculate the first 50 principal components (PCs) [24] using the *prcom_irlba* function from the ilrba R package [25]. This lower dimensional space is used to identify MNNs, and in the clustering and fuzzy correction steps. However, during the calculation of the correction vector step, we use the original input datasets.

We calculate mutual nearest neighbors (MNN) pairs [5] using 50 PCs to obtain batch effect observations. We assume that at least one cell is shared between the batches to integrate. The MNN pairs are defined by the intersection of the *crossed k nearest neighbors* for each cell of two input batches. For example, for a cell $c_1$from batch one, we find the k closest cells from batch two, and for cell $c_2$from batch two, we find the k closest cells from batch one. If c_1 and $c_2$are mutually contained on each other's nearest neighbor set, they are considered as a MNN pair. In Canek, to identify MNN pairs we first find the crossed 30 nearest neighbors of the query and reference batches using the *get.knn* function from the FNN R package [26]. We then select those cells that fulfill the MNN criteria to form cell pairs. We treat the gene expression differences from these pairs as observations of the batch effect.

Clustering

Following Haghverdi et al. (2018) [5] we assume that the batch effect is almost orthogonal to the biological space, and that the variations due to the batch effect are smaller than the biological variation (see Supplementary material of [5] for a deeper discussion of these assumptions). Small variations to this orthogonality assumption can be caused by noise or by non-linearities. A common way to deal with non-linear dynamics is to linearize over bounded regions [27], to solve each of these local problems, and, if necessary, to join all the pieces back into a non-linear global solution. Following this idea, we partition the query batch into memberships, which we define as a bounded set of related cells, using the Louvain algorithm implemented in the igraph R package [28]. By default, clustering is done using the first 10 PCs.

Correction Vector

Following our *local orthogonal batch effect* assumption*, for each membership* we state the relation:

$$g^i_{Q_k} = g^i_{R_k} + g^i_{BE_k} + \epsilon$$

where $g^i, i = 1, \dots, n$, is the log-normalized gene expression level of the n genes from the input batches. The batch effect g_{BE_k} is represented as an additive value in the query batch g_{Q_k} in terms of the same gene in the reference batch g_{R_k}, $k = 1, \dots, p$, being p the number of MNN pairs from the membership under analysis. Finally, ϵ represents a normally distributed random error term with mean zero and standard deviation σ, which we assume to be independent of g^i on each membership. Thus, using

$$g_{Q_k} - g_{R_k} = g_{BE_k} + \epsilon$$

on each gene i, the term $g_{BE_k} + \epsilon$ would be normally distributed with mean $\mu = g_{BE}$ and standard deviation σ. Accordingly, a good estimation of the batch effect would be the mean of the gene expression subtraction between MNN cells pairs (e.g. $\hat{g}_{BE} = \frac{1}{n}\sum_{k=1}^{p}(g_{Q_k} - g_{R_k})$). But there is a complication with this approach, since *erroneous pairs* between cells from distinct but related cell types could be formed, resulting in the incorrect integration of dissimilar subpopulations [6]. To tackle this problem, reasoning that abnormal pairs would appear as outliers to the normal distribution of $g_{BE_k} + \epsilon$, we estimate a correction vector

$$CV = -\begin{bmatrix} \hat{g}^1_{BE_k} = Med(g^1_{Q_k} - g^1_{R_k}) \\ \vdots \\ \hat{g}^n_{BE_k} = Med(g^n_{Q_k} - g^n_{R_k}) \end{bmatrix}$$

where the function Med represents the statistical median, which is less affected by outliers than the mean. Canek uses this approach by default to reduce the impact from outliers, but it is possible to perform an optional filtering step (with extra computational cost) based on the interquartile range to detect MNN outliers (see *Filtering* section).

Fuzzy correction

From the steps described above, each cell from the same membership will be assigned the same correction vector. We use fuzzy logic to smoothly join the membership-specific corrections into a cell-specific one, where each cell has a unique correction vector (see Supplementary Figure 5).

Using the PCs of the query batch, we create a minimum spanning tree (MST) over the memberships' center points (MCs) using the *mst* function from the R package igraph [28] (Supplementary Figure 5a,b). For each edge of the MST, we construct a pair of membership functions (MFs). These MFs are used to calculate a fuzzy score for the cells (Supplementary Figure 5c,d). For example, let us consider an edge that joins the centers of memberships number 1 (MC_1) and 2 (MC_2). For each cell j that belongs to memberships 1 or 2, we define the vector V_j as a vector for cell j from MC_1 in the PCs embeddings. Similarly, let V_{MC_2} be the vector corresponding to MC_2. Then, we obtain the scalar projection p_j for each cell j onto the line connecting MC_1 and MC_2 as:

$$p_j = V_j \cdot \frac{V_{MC_2}}{\|V_{MC_2}\|}$$

where the operator $\cdot$ denots the dot product, and $\|V_{MC_2}\|$ is the length of V_{MC_2}. We then construct the MFs (i.e., MF_1 and MF_2) as

$$MF_1(j) = 1 - \frac{p_j - p_{min}}{p_{max} - p_{min}}$$

$$MF_2(j) = \frac{p_j - p_{min}}{p_{max} - p_{min}}$$

Here, p_{max} and p_{min} are the maximum and the minimum of the scalar projections of the cells in the memberships ($p_{max} = \max_j p_j$ and $p_{min} = \min_j p_j$). In this way, the membership function MF_1 (MF_2) takes the maximum value 1 (the minimum value 0) for p_{min} and the minimum value 0 (the maximum value 1) for p_{max}, respectively, and linearly interpolates for the other values of the projections. (Supplementary Figure 6).

We calculate cell specific correction vectors CV_j by using the Takagi-Sugeno approach [29] to combine the membership's correction vector $CV^{(l)}$ (see *Correction Vector* section) with the membership functions:

$$CV_j = \frac{\sum_l MF_l(j) CV^{(l)}}{\sum_l MF_l(j)}$$

When a membership l is connected to several edges, we use the average of the membership functions MF_l defined for all the edges associated with membership l.

Even though the fuzzy scheme is applied in a low dimensional representation, the final output is in the original dimensionality of the input datasets. We recommend using Canek with the fuzzy step, but to skip it users can set the boolean parameter *fuzzy to FALSE*. In this case the final integration will be done using a membership-specific correction instead of a cell-specific one.

Hierarchical integration

We define a hierarchical integration when Canek is applied to more than two input batches. We first sort the batches by cell number in descending order and use the batch with the higher number of cells as the first reference batch. To determine the query batch, we prioritize to integrate first related batches as they would have a higher number of MNN pairs. The query batch is therefore chosen as the batch sharing the highest number of MNN pairs with the

reference. For this, we obtain their first three PCs using the prcomp_irlba function in the irlba R package [25], find the MNN pairs and select the query batch as the one with the highest number of pairs. Once the reference and the selected query batch are integrated, we define the integrated batch as the new reference, and again select the query batch following the same procedure as before. We continue this process until all the input batches are integrated. The hierarchical integration is optional and can be deactivated by setting the boolean parameter *hierarchical* to FALSE. In this case, the order of integration follows the order in the input list.

Filtering

We assume that erroneous MNN pairs would appear as outliers from the normal distribution of $g_{BE_k} + \epsilon = g^n_{Q_k} - g^n_{R_k}$ (see *Correction Vector* section). We use the median function to reduce the impact of these outliers on the correction vector estimation. In addition, the user can select an extra filtering step based on the interquartile range:

$$IQR = Q_{75} - Q_{25}$$

where Q_{75} and Q_{25} are the 75th and 25th percentiles of the distribution of the p MNN pairs' Euclidean distance $d(k), k = 1, \dots, p$. Therefore, we will select and filter any outlier MNN pairs as:

$$MNN_k \; \boldsymbol{IS} \; outlier \; \boldsymbol{IF} \; (d(MNN_k) < (Q_{25} - 1.5IQR) \mid d(MNN_k) > (Q_{75} + 1.5IQR)).$$

Analysis details

Data pre-processing

We performed the same data pre-processing for all the analyses. We implemented the "Standard Workflow" from the Seurat R package [13], which involves:

- **Normalization**: using the function NormalizeData. Gene expression levels are divided by the total number of transcripts and multiplied by 10,000. The results are then log normalized.
- **Identification of high variable features**: using the function FindVariableFeatures. Genes that show high variations among cells are selected using the vst method.

Batch-correction algorithms

Table 1 lists the batch correction methods used.

Method	Batch effect corrected output	Package version
Seurat	Normalized gene expression matrix	Seurat version 3.2.2 [6]
MNN	Normalized gene expression matrix	Bioconductor's batchelor version 1.6.2 [5]
Scanorama	Normalized gene expression matrix	Scanorama version 1.6 [8]
ComBat	Normalized gene expression matrix	sva version 3.38.0 [30]
Harmony	Normalized feature reduction vectors	Harmony version 1.0 [9]
Liger	Normalized feature reduction vectors	Liger version 0.5.0 [10]
ComBat-seq	Normalized gene expression matrix	sva version 3.38.0 [31]
scMerge	Normalized gene expression matrix	scMerge version 1.6.0 [14]
Canek	Normalized gene expression matrix	Canek version 0.1.7

Table 1. Batch effect correction methods

To objectively compare the batch effect correction methods, we used the same pre-processed data and the same variable genes on each method. We obtained the variable genes from

Seurat's integration using the *VariableFeatures(assay="integrated")* function from the Seurat R package [13] and used them to subset the pre-processed uncorrected datasets. We implemented the integration methods as follows:

Seurat. We used the *FindIntegrationAnchors* and *IntegrateData* functions with default parameters from the Seurat R package [13].

Canek. We used the *RunCanek* function from the Canek R package with default parameters.

MNN. We used the *mnnCorrect(cos.norm.out = FALSE)* function with default parameters from the batchelor Bioconductor package [5].

Scanorama. We used the *scanorama.correct(return_dense=TRUE)* with default parameters from the scanorama Python library [8].

ComBat. We used the *ComBat* function with default parameters from the sva R package [30].

Harmony. We used the *RunHarmony* function with default parameters from the harmony R package [9].

Liger. We used the *RunOptimizeALS(k=20, lambda=5)* and the *RunQuantileNorm* functions with default parameters from the SeuratWrappers R package [10].

ComBat-seq. We used the *ComBat_seq* function with default parameters from the sva R package [31].

scMerge. We used the *scMerge(k=3)* function with default parameters from the scMerge R package [14].

After correcting batch effects, we scaled each of the integrated and uncorrected datasets using the *ScaleData* function from the Seurat R package, except for Harmony and Liger integrations, as their output is already a low dimensional space.

Dimensionality reduction

We obtained the principal components [24] from the corrected and uncorrected datasets using the RunPCA function from the Seurat R package. We used the first 10 PCs as the standard in all the tests. In the case of the UMAP representation [12], we applied the *RunUMAP* from the Seurat R package to the selected PCs.

Simulated data

We used the *splatSimulate* function from the splatter Bioconductor package [16] to simulate three batches with batch effect. Splatter allows us to simulate cell types whether as groups or paths. Because we wanted to assess cell type preservation on a mixed population scenario with clearly defined groups along with a differentiation process we simulated paths and groups separately and merged them. Then, we manually removed cells such that the batches shared only one cell type. The final cell type composition is:

Method	Path cells		Group cells		
	Cell-1	Cell-2	Cell-3	Cell-4	Cell-5

Batch-1	✓	✓	✓	-	-
Batch-2	-	✓	-	✓	-
Batch-3	✓	-	-	-	✓

Table 2. Cell type distribution on simulated data

After removing the cell types, the number of cells per batch is: Batch 1 – 1,671 cells, Batch 2 – 975 cells, and Batch 3 – 964 cells. All of them with the same 2,000 genes.

To obtain the gold standard without batch effects, we used *splatSimulate(batch.rmEffect = TRUE)* and removed the same cells as the simulations with batch effects.

Running time benchmark

We used the *splatSimulate* function from the *Splatter* Bioconductor package [16] to simulate two datasets with a 2,000 genes and a varying number of cells in the range of 10k to 100k. Each dataset contained three cell types with appearance probabilities of 0.3, 0.3, and 0.4 respectively. We applied each of the correction methods five times on each of the simulated datasets and recorded the time. We used the *geom_smooth* function from the ggplot2 R package [32] to plot the time trend lines.

Public datasets

Table 3 lists the public datasets we used.

Dataset	Number of cells	Technology	Publication
Jurkat cells	3,258	10x	Zheng. G.X. el al. [11]
HEK293T	2,885		
Jurkat:HEK293T 50:50 mixture	3,388		
Spleen	1,697	SMART-seq2	Tabula Muris [15]
	9,552	10x	
Pancreatic	8,569	inDrop	Baron, M.,et al. [19]
	2,285	CEL-seq	Muraro, M.J., et al. [17]
	1,004	CEL-seq2	Grün, D., et al. [21]
	638	Fluidgm C1	Lawlor, N., et al. [18]
	2,394	Smart-seq2	Segerstolpe, A., et al. [20]

Table 3. Public datasets.

Jurkat/t293 data analysis

We used the following publicly available datasets:

- 293T cells. https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/293t_3t3
- Jurkat cells. https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/jurkat
- 50:50 Jurkat:293T cell mixture. https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/jurkat:293t_50:50

In the 50:50 Jurkat:293T cell dataset, we used the *kmeans* function from the *stats* R package [33] with k = 2, checked the expression of the XIST and CD3D3 genes and assigned the appropriate cell type labels.

Spleen data analysis (pseudo-batch)

- We use the publicly available dataset from Tabula Muris [15]. https://ndownloader.figshare.com/files/13090478

Pancreatic data analysis

- We obtained the five public datasets [17-21] from the SeuratData R package[6] and used the provided cell type labels.

PBMC unstimulated and IFN-β-stimulated data analysis

- We obtained the public datasets [22] from the SeuratData R package [6] and used the provided cell type labels.

Metrics

We evaluated the results from the batch-correction methods by scoring the mixing between batches with the k-nearest-neighbor batch effect test (kBET) [34], and the cell purity preservation with the average silhouette width (Silhouette) [35]. We used the kBET and batch_sil functions from the kBET R package [34].

The kBET metric provides a rejection rate within 0 and 1 after testing batch mixing at the local level. The kBET's score could be affected by the choice in the number of k-nearest neighbors (kNN). To objectively assess the different integration methods, following the idea of Tran et al. [4], we obtained the mean cell number of the datasets and performed the scoring by fixing the kNN size as the 5%, 15%, and 30% of this mean. To ease the interpretation of this metric, we calculated an "acceptance rate" by subtracting the rejection rate from 1.

We used the silhouette coefficient to assess cell purity after integration. This metric analyzes the separation among cells from the same cluster as compared with cells from other clusters [35]. Let $a(i)$ be the average Euclidean distance of cell i to all other cells from the same cluster as i, then the silhouette width $s(i)$ is defined as:

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

where $b(i)$ is calculated as

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

being $d(i, C)$ the average distance of cell i to all the other cells assigned to different clusters C. A higher score means a longer separation between clusters and a lower score means a shorter separation. We used the cell type labels provided for each dataset as inputs to the silhouette coefficient, except on the pseudo-batch experiment, where we obtained the cluster labels using the *FindNeighbors* and *FindClusters(resolution = 0.5)* functions from the Seurat R package.

Data availability

Code availability

Canek is implemented as an R package and is available from GitHub (https://github.com/MartinLoza/Canek).

Acknowledgements

We thank Dr. Alexis Vandenbon for valuable comments on our manuscript.

Author contributions

ML conceived this work with contributions from ST and DD. ML and DD implemented the methods. ML performed the analyses with contributions from DD. ML and DD wrote the manuscript with contributions from ST. All authors edited and approved the submitted manuscript.

Competing interests

The authors declare no competing interests.

References

1. Svensson, V., R. Vento-Tormo, and S.A. Teichmann, *Exponential scaling of single-cell RNA-seq in the past decade.* Nat Protoc, 2018. **13**(4): p. 599-604.
2. Goh, W.W.B., W. Wang, and L. Wong, *Why Batch Effects Matter in Omics Data, and How to Avoid Them.* Trends Biotechnol, 2017. **35**(6): p. 498-507.
3. Leek, J.T., et al., *Tackling the widespread and critical impact of batch effects in high-throughput data.* Nat Rev Genet, 2010. **11**(10): p. 733-9.
4. Tran, H.T.N., et al., *A benchmark of batch-effect correction methods for single-cell RNA sequencing data.* Genome Biol, 2020. **21**(1): p. 12.
5. Haghverdi, L., et al., *Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors.* Nat Biotechnol, 2018. **36**(5): p. 421-427.
6. Stuart, T., et al., *Comprehensive Integration of Single-Cell Data.* Cell, 2019. **177**(7): p. 1888-1902.e21.
7. Polański, K., et al., *BBKNN: fast batch alignment of single cell transcriptomes.* Bioinformatics, 2020. **36**(3): p. 964-965.
8. Hie, B., B. Bryson, and B. Berger, *Efficient integration of heterogeneous single-cell transcriptomes using Scanorama.* Nat Biotechnol, 2019. **37**(6): p. 685-691.
9. Korsunsky, I., et al., *Fast, sensitive and accurate integration of single-cell data with Harmony.* Nat Methods, 2019. **16**(12): p. 1289-1296.
10. Welch, J.D., et al., *Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity.* Cell, 2019. **177**(7): p. 1873-1887.e17.
11. Zheng, G.X., et al., *Massively parallel digital transcriptional profiling of single cells.* Nat Commun, 2017. **8**: p. 14049.
12. McInnes, L., J. Healy, and J. Melville, *Umap: Uniform manifold approximation and projection for dimension reduction.* arXiv preprint arXiv:1802.03426, 2018.
13. Butler, A., et al., *Integrating single-cell transcriptomic data across different conditions, technologies, and species.* Nat Biotechnol, 2018. **36**(5): p. 411-420.
14. Lin, Y., et al., *scMerge leverages factor analysis, stable expression, and pseudoreplication to merge multiple single-cell RNA-seq datasets.* Proc Natl Acad Sci U S A, 2019. **116**(20): p. 9775-9784.
15. Tabula Muris, C., et al., *Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris.* Nature, 2018. **562**(7727): p. 367-372.

16. Zappia, L., B. Phipson, and A. Oshlack, *Splatter: simulation of single-cell RNA sequencing data.* Genome Biol, 2017. **18**(1): p. 174.
17. Muraro, M.J., et al., *A Single-Cell Transcriptome Atlas of the Human Pancreas.* Cell Syst, 2016. **3**(4): p. 385-394 e3.
18. Lawlor, N., et al., *Single-cell transcriptomes identify human islet cell signatures and reveal cell-type-specific expression changes in type 2 diabetes.* Genome Res, 2017. **27**(2): p. 208-222.
19. Baron, M., et al., *A Single-Cell Transcriptomic Map of the Human and Mouse Pancreas Reveals Inter- and Intra-cell Population Structure.* Cell Syst, 2016. **3**(4): p. 346-360.e4.
20. Segerstolpe, A., et al., *Single-Cell Transcriptome Profiling of Human Pancreatic Islets in Health and Type 2 Diabetes.* Cell Metab, 2016. **24**(4): p. 593-607.
21. Grün, D., et al., *De Novo Prediction of Stem Cell Identity using Single-Cell Transcriptome Data.* Cell Stem Cell, 2016. **19**(2): p. 266-277.
22. Kang, H.M., et al., *Multiplexed droplet single-cell RNA-sequencing using natural genetic variation.* Nature biotechnology, 2018. **36**(1): p. 89.
23. Henig, N., et al., *Interferon-beta induces distinct gene expression response patterns in human monocytes versus T cells.* PLoS One, 2013. **8**(4): p. e62366.
24. Pearson, K., *LIII. On lines and planes of closest fit to systems of points in space.* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1901. **2**(11): p. 559-572.
25. Lewis, B.W., J. Baglama, and L. Reichel, *The irlba Package.* 2019.
26. Beygelzimer, A., et al., *Package 'FNN'.* Accessed June, 2015. **1**.
27. Strogatz, S., *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering (studies in nonlinearity).* 2001.
28. Csardi, G. and T. Nepusz, *The igraph software package for complex network research.* InterJournal, complex systems, 2006. **1695**(5): p. 1-9.
29. Takagi, T. and M. Sugeno, *Fuzzy identification of systems and its applications to modeling and control.* IEEE transactions on systems, man, and cybernetics, 1985(1): p. 116-132.
30. Johnson, W.E., C. Li, and A. Rabinovic, *Adjusting batch effects in microarray expression data using empirical Bayes methods.* Biostatistics, 2007. **8**(1): p. 118-27.
31. Zhang, Y., G. Parmigiani, and W.E. Johnson, *: batch effect adjustment for RNA-seq count data.* NAR Genom Bioinform, 2020. **2**(3): p. lqaa078.
32. Wickham, H., W. Chang, and M.H. Wickham, *Package 'ggplot2'.* Create Elegant Data Visualisations Using the Grammar of Graphics. Version, 2016. **2**(1): p. 1-189.
33. Team, R.C., *Vienna: R Foundation for Statistical Computing, 2020.* 2020.
34. Büttner, M., et al., *A test metric for assessing single-cell RNA-seq batch correction.* Nat Methods, 2019. **16**(1): p. 43-49.
35. Rousseeuw, P.J., *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis.* Journal of computational and applied mathematics, 1987. **20**: p. 53-65.