A joint deep learning model enables simultaneous batch effect correction, denoising and clustering in single-cell transcriptomics

Justin Lakkis, David Wang, Yuanchao Zhang, et al.

Genome Res. published online May 25, 2021
Access the most recent version at doi:10.1101/gr.271874.120

A joint deep learning model enables simultaneous batch effect correction, denoising and clustering in single-cell transcriptomics

Justin Lakkis[1,*], David Wang[2], Yuanchao Zhang[1], Gang Hu[3], Kui Wang[4], Huize Pan[5], Lyle Ungar[6], Muredach P. Reilly[5], Xiangjie Li[7,*], Mingyao Li[1,*]

1. Department of Biostatistics, Epidemiology and Informatics, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA.

2. Graduate Group in Genomics and Computational Biology, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA.

3. School of Statistics and Data Science, Key Laboratory for Medical Data Analysis and Statistical Research of Tianjin, Nankai University, Tianjin 300071, China.

4. Department of Information Theory and Data Science, School of Mathematical Sciences and LPMC, Nankai University, Tianjin 300071, China.

5. Division of Cardiology, Department of Medicine, Columbia University Irving Medical Center, New York, NY 10032, USA.

6. Department of Computer and Information Science, School of Engineering and Applied Sciences, University of Pennsylvania, Philadelphia, PA 19104, USA.

7. State Key Laboratory of Cardiovascular Disease, Fuwai Hospital, National Center for Cardiovascular Diseases, Chinese Academy of Medical Sciences and Peking Union Medical College, Beijing 100037, China.

***Correspondence:**

Justin Lakkis (jlakkis@pennmedicine.upenn.edu)

Xiangjie Li (xiangjie631@outlook.com)

Mingyao Li, Ph.D. (mingyao@pennmedicine.upenn.edu)

Abstract

Recent developments of single-cell RNA-seq (scRNA-seq) technologies have led to enormous biological discoveries. As the scale of scRNA-seq studies increases, a major challenge in analysis is batch effects, which are inevitable in studies involving human tissues. Most existing methods remove batch effects in a low-dimensional embedding space. Although useful for clustering, batch effects are still present in the gene expression space, leaving downstream gene-level analysis susceptible to batch effects. Recent studies have shown that batch effect correction in the gene expression space is much harder than in the embedding space. Methods such as Seurat 3.0 rely on the mutual nearest neighbor (MNN) approach to remove batch effects in gene expression, but MNN can only analyze two batches at a time, and it becomes computationally infeasible when the number of batches is large. Here we present CarDEC, a joint deep learning model that simultaneously clusters and denoises scRNA-seq data while correcting batch effects both in the embedding and the gene expression space. Comprehensive evaluations spanning different species and tissues showed that CarDEC outperforms Scanorama, DCA + Combat, scVI, and MNN. With CarDEC denoising, non-highly variable genes offer as much signal for clustering as the highly variable genes (HVGs), suggesting that CarDEC substantially boosted information content in scRNA-seq. We also showed that trajectory analysis using CarDEC's denoised and batch corrected expression as input revealed marker genes and transcription factors that are otherwise obscured in the presence of batch effects. CarDEC is computationally fast, making it a desirable tool for large-scale scRNA-seq studies.

Introduction

Single-cell RNA sequencing (scRNA-seq) analysis has substantially advanced our understanding of cellular heterogeneity and transformed biomedical research. However, the analysis of scRNA-seq data remains confounded by batch effects, which are inevitable in analyses of human tissue and are prevalent in many scRNA-seq studies in general (Hicks et al. 2018; Lahnemann et al. 2020). Several methods have been developed to remove batch effects in scRNA-seq data analysis (Haghverdi et al. 2018; Lopez et al. 2018; Barkas et al. 2019; Korsunsky et al. 2019; Stuart et al. 2019b; Welch et al. 2019; Li et al. 2020; Polanski et al. 2020). These methods can be divided into two categories: 1) batch correction in the low-dimensional embedding space, and 2) batch correction in the original gene expression space. Most published papers belong to the first category (Barkas et al. 2019; Korsunsky et al. 2019; Welch et al. 2019; Li et al. 2020; Polanski et al. 2020). Although useful for profiling the overall characteristics of cells such as clustering and trajectory reconstruction, these methods cannot be used for downstream gene-level analysis like differential expression and co-expression analysis.

A recent benchmarking study has shown that correcting batch effects in the gene expression space is much more challenging than in the embedding space (Lucken et al. 2020). Popular methods such as Seurat 3.0 (Stuart et al. 2019b) rely on the mutual nearest neighbor (MNN) approach (Haghverdi et al. 2018) to remove batch effects in the gene expression space, but MNN can only analyze two batches at a time. Its performance is affected by the order in which batches are corrected and it quickly becomes computationally infeasible when the number of batches gets large. Moreover, our evaluations indicate that MNN performs poorly for removing batch effects for genes that are not highly variable. Non-highly variable genes represent the majority of genes in the genome, where batch effects constitute a larger fraction of variance in the transcriptome and are much harder to correct.

To address this gap in the literature, we present CarDEC (**C**ount **a**dapted **r**egularized **D**eep **E**mbedded **C**lustering), a joint deep learning framework for simultaneous batch effect correction, denoising, and clustering of scRNA-seq data. Rather than explicitly modeling batch effect, CarDEC jointly optimizes its reconstruction loss with a self-supervised clustering loss. By minimizing a clustering loss iteratively, the batch effects in the embedding are reduced and cell type signal is improved (Li et al. 2020). The denoised gene expression values, computed from this embedding using a decoder, are then corrected for batch effects as well. To address the difficulty of batch correcting genes that are not highly variable, which suffer from a lower cell type signal-to-noise ratio, we designed CarDEC using a branching architecture that treats highly variable genes (HVGs) and the remaining genes, which we designate as lowly variable genes (LVGs), as distinct feature blocks.

CarDEC is unique among batch effect correction methods in that it implicitly corrects for batch effects through joint optimization of its dual objective function, rather than explicitly modeling batch effects using batch indicators as in methods such as MNN (Haghverdi et al. 2018) and scVI (Lopez et al. 2018). Moreover, it corrects batch effects both in the low-dimensional embedding space and the original gene expression space. CarDEC's architecture is founded on the idea of treating HVGs and LVGs as different "feature blocks," which enables CarDEC to use the HVGs to drive the clustering loss, while still allowing the LVG reconstructions to depend on the rich, batch corrected embedding learned from the HVGs, to help remove batch effects in the LVGs.

Results

Overview of CarDEC and evaluation

An outline of the CarDEC workflow is shown in Fig. 1. CarDEC starts with data preprocessing and pretraining of an autoencoder using HVGs with a mean squared error reconstruction loss function. After

pretraining, the weights learned from the pretrained autoencoder are transferred over to the main CarDEC model, which treats HVGs and LVGs as different feature blocks. The main CarDEC loss function is a weighted combination of the reconstruction losses for the HVGs and the LVGs, and a self-supervised clustering loss function driven by the HVGs. This combined loss function allows CarDEC to preserve local structure of the data during clustering (Guo et al. 2017). By minimizing this self-supervised combined loss function, CarDEC not only improves the low-dimensional embedding for clustering but the reconstructed genewise features, which are computed as a function of the low-dimensional embedding, are also denoised and batch effect corrected, leading to improved gene expression quality.

We evaluated CarDEC on a diverse set of challenging real datasets that range from human to mouse and have different flavors of batch effects. In our evaluations, we wish to assess two properties of CarDEC: 1) its ability to recover biological signals in the data, and 2) its ability to remove spurious technical signals driven by batch effects. An ideal method should strive to remove batch effects while maintaining true biological variations. We compared CarDEC with several state-of-the-art scRNA-seq methods for denoising, batch effect correction, and clustering. scVI (Lopez et al. 2018) and DCA (Eraslan et al. 2019) are multi-use methods that provide denoised counts in the gene expression feature space, and also a low-dimensional embedding that can be used for tasks like clustering and visualization. scVI also attempts to correct for batch effects by conditioning on batch annotation when modeling the denoised counts with a zero-inflated negative binomial distribution. Since DCA is not designed for batch effect correction, to make a fair comparison, we applied Combat (Johnson et al. 2007) to DCA denoised gene expression. MNN (Haghverdi et al. 2018) is a batch correction method that merges batches in a pairwise manner and generates batch corrected gene expression on a cosine scale. We used the implementation of the MNN method provided in the R programming language (R Core Team 2020). Scanorama (Hie et al. 2019) is also based on the mutual nearest neighbor idea, but it finds matching elements among all

batches at once, thus speeding it up computationally and making the method invariant to batch order. scDeepCluster (Tian et al. 2019) is a clustering method that also draws inspiration from the self-supervised clustering loss (Guo et al. 2017). We provide the exact software implementations of these packages that we used in Supplemental Table 3.

To measure the degree of batch mixing, we examined the batchwise centroids after denoising and/or batch correction with each method by calculating a coefficient of variation (CV) metric. For each gene, a CV is calculated for a given cell type using the centroid of each batch in that cell type. Then, to obtain a single CV score, we take the weighted average of the cell-type-specific CVs, where the weight of a cell type is the fraction of the dataset's cells that belong to that cell type. The reason for computing CV scores within cell types is that we expect minimal biological heterogeneity within cell types, so in the absence of batch effects we expect batch centroids to be similar to one another within, but not between cell types. A higher value of CV corresponds to greater variation of gene expression among batches and less batch mixing, whereas a good batch effect removal method should drive the CV value close to zero.

Application to human pancreatic islet data from four protocols

A unique feature of CarDEC is the branching architecture for both the HVGs and the LVGs. To demonstrate that this architecture is key in removing batch effects, we combined four datasets consisting of scRNA-seq expression data in human pancreas generated using Fluidigm C1 (Lawlor et al. 2017), Smart-seq2 (Segerstolpe et al. 2016), CEL-Seq (Grun et al. 2016), and CEL-Seq2 (Muraro et al. 2016). This is a challenging task as there are strong batch differences among these different scRNA-seq protocols, and analysis using the raw data as input yielded low Adjusted Rand Indexes (ARIs) (Supplemental Fig. S1). The branching architecture was designed with two objectives in mind. First, we wish to show that when correcting batch effects and denoising both the HVGs and the LVGs, using a

branching model that treats these feature blocks differently improves the quality of denoised expression values relative to a naïve architecture that treats these feature blocks the same. Second, we hope to design a model architecture such that including the LVGs in the model does not worsen denoising and batch effect correction quality of the HVGs, relative to a naïve model that only denoises the HVGs and does not attempt to denoise LVGs.

As shown in Fig. 2, the branching architecture posts significant performance boosts over the naïve architecture that treated all genes as the same feature block in the input. The branching architecture performed better for denoising both the HVGs (ARI of 0.93 over 0.72) and the LVGs (ARI of 0.83 over 0.67) relative to the naïve architecture (Fig. 2A,B), underscoring the necessity of using the branching architecture to denoise all genes as efficiently as possible. We also observed that for the purpose of denoising and batch correcting only the HVGs, the branching architecture performed just as well as a naïve model that only included the HVGs and completely discarded the LVGs (ARI of 0.93 vs 0.95) (Supplemental Fig. S2). This verifies that the branching architecture does not trade off denoising and batch correction effectiveness on the HVGs at all to denoise the LVGs. Additionally, the denoised counts from CarDEC showed less batch effects compared to denoised expression from Scanorama, scVI, and batch corrected expression from MNN (Fig. 2A,B). Fig. 2C shows that genewise CV scores obtained from CarDEC are the closest to zero among all compared methods. We also noticed that the clustering accuracy obtained when clustering is done using denoised values in the gene expression space is similar to the clustering accuracy obtained when using the embedding of CarDEC to do clustering (Fig. 2D).

Application to macaque retina data with multi-level batch effect

After finalizing the CarDEC architecture, we next evaluated the performance of CarDEC on a macaque retina dataset (Peng et al. 2019). This dataset poses a great challenge for batch effect correction and

denoising because it features strong, multi-level batch effects, with cells sequenced from two different regions, four different macaques, and thirty different samples (Supplemental Fig. S3).

For the task of denoising and batch effect correction in the gene expression space, CarDEC was again the best performing method (Fig. 3, Supplemental Fig. S4, Supplemental Fig. S5), followed by scVI whose ARIs are close to CarDEC. CarDEC and scVI not only removed the multi-level batch effects but also preserved inter-cell type variation. Notably, the ARI for clustering using the LVG denoised and batch effect corrected counts from CarDEC is 0.98 and is 0.97 for scVI (Fig. 3B), which is as high as that when using the HVGs to do clustering (Fig. 3A). As a comparison, the ARI is only 0.15 using the LVG raw counts as input for clustering (Supplemental Fig. S3). This suggests that the denoising and batch correction in CarDEC and scVI substantially boosted the signal-to-noise ratio in the LVGs. Moreover, CarDEC's and scVI's genewise CVs are the closest to zero, providing evidence that cells were mixed well by batch by these two methods (Fig. 3C, Supplemental Fig. S6).

The other methods all struggled with batch effects in the denoised counts. Scanorama largely failed to correct batch effects: when using Scanorama batch corrected gene expression as input, the cells were separated primarily by batch rather than by cell type. For the LVGs, its ARI is as low as that when using the LVG raw counts as input for clustering (Scanorama 0.21 vs raw 0.15) (Fig. 3B, Supplemental Fig. S3). DCA had slightly higher ARIs than Scanorama for both the HVGs and the LVGs, although both are significantly lower than CarDEC and scVI (Fig. 3A,B). MNN performed much better than Scanorama and DCA for batch correcting the HVG counts, achieving an ARI of 0.91 (Fig. 3A). However, it still fell short of CarDEC and scVI for this evaluation (CarDEC ARI 0.98 and scVI ARI 0.96). Looking more closely at the HVG UMAP plots, the batches were mixed less thoroughly with MNN than they were for CarDEC and scVI, and the cells were separated less by cell type indicating that MNN failed to completely recover cell

type variation. This is further confirmed by the genewise CV density plot in which the MNN density curve is further away from zero than CarDEC and scVI (Fig. 3C). For removing batch effects in the LVG counts, MNN was the worst performing method because it removed nearly all biological variations, leaving only batch effects.

For the simpler task of clustering using the low-dimensional embedding representation, existing methods did considerably better than they did at batch effect correction in the gene expression space, but still fell short of CarDEC and scVI (Fig. 3D, Supplemental Fig. S7). CarDEC and scVI achieved ARIs of nearly 1 for clustering using the embedding. DCA and scDeepCluster performed better than Louvain's algorithm using raw HVGs, but still fell short of achieving as high of an ARI as CarDEC and scVI. Scanorama struggled on this dataset with an ARI of only 0.57.

Application to mouse cortex and PBMC data from four protocols

We next compared different methods using a mouse cortex dataset (Ding et al. 2020). This dataset poses a great challenge for batch correction and denoising on two fronts (Supplemental Fig. S9). First, it exhibits very serious batch effects because cells were generated using four different scRNA-seq protocols. Furthermore, this dataset is heavily dominated by excitatory and inhibitory neurons, and the other cell types are rare, so preserving biological variation is especially imperative for detecting and analyzing these rarer subpopulations.

For the task of denoising and batch correcting the gene expression space, CarDEC and scVI performed considerably better than the other methods (Fig. 4). CarDEC performed the best, followed closely by scVI, at balancing between removing batch effects while preserving as much cell type variability as possible. For both CarDEC and scVI, the ARIs are similar when using the HVG denoised counts and the

LVG denoised counts as input for clustering (Fig. 4A,B). As a comparison, the ARIs are only 0.28 and 0.26 when using the HVG and the LVG raw counts as input for clustering, respectively (Supplemental Fig. S9). The relatively low ARI when using the HVG raw counts as input for clustering demonstrates the strong batch effects in this dataset. However, for this challenging dataset, using CarDEC denoised and batch corrected LVG counts, the ARI increased to 0.78, suggesting that CarDEC substantially boosted the signal-to-noise ratio in the LVGs by simultaneous denoising and batch effect removal. We observed a similar signal boost by scVI in which the ARI for the LVGs is 0.73. The genewise CVs for CarDEC are also the closest to zero among all methods (Fig. 4C). Although scVI achieved high ARIs for both the HVG and LVG denoised gene expression, the cells appeared to be less well mixed than CarDEC (Fig. 4A,B), which is in agreement with its larger CVs than CarDEC.

DCA largely failed for this dataset (Fig. 4A,B), even though its denoised gene expression was also batch corrected by Combat. For both the HVGs and the LVGs, DCA separated the cells purely by scRNA-seq protocol with no mixing of cells from different batches. After denoising using DCA and batch correction with Combat, cell variation was driven entirely by batch, rendering the denoised counts ineffective for downstream analyses. For batch correcting the HVGs, Scanorama was the third-best performer (Fig. 4A), followed by MNN. MNN did not merge batches to the extent that CarDEC did and failed to preserve as much cell type variability, causing cell types to mix more. For removing batch effects in the LVGs, MNN did considerably worse than CarDEC and scVI (Fig. 4B). It suffered from the same problems as DCA for the LVGs in that cell type variation was lost and all variability was driven by batch. We also noticed that the CVs for DCA are the closest to zero, however, the small CVs are mainly due to the overcorrection of Combat as the ARIs are low for both the HVGs and the LVGs.

Due to the strong batch effects in this dataset, even the simpler task of clustering using embedding was very difficult (Fig. 4D, Supplemental Fig. S10). In particular, scDeepCluster performed poorly at this task, scoring a lower ARI than a straightforward application of Louvain's algorithm to the raw data. The ARI for Scanorama is only slightly better than that obtained from the raw data. For this task, scVI was the leader, achieving an ARI of 0.74, followed closely by CarDEC with an ARI of 0.73.

We also analyzed a dataset of human PBMCs from the same paper (Ding et al. 2020) as the mouse cortex data. This dataset was similar to the cortex dataset: featuring eight batches spanning five scRNA-seq protocols and the results were largely the same: CarDEC and scVI were the best performers for denoising/batch correcting the HVGs and also the best for denoising/batch correcting the LVGs (Supplemental Fig. S12-S14).

Application to human monocyte data with pseudotemporal structure

We next show the utility of CarDEC for improving trajectory analysis for cells with pseudotemporal structure. We analyzed an scRNA-seq dataset generated from monocytes derived from human peripheral blood mononuclear cells by Ficoll separation followed by CD14-› and CD16-positive cell selection (Li et al. 2020). This dataset includes 10,878 monocytes from one healthy subject. The cells were processed in three batches from blood drawn on three different days. Although monocytes can be classified as classical ($CD14^{++}/CD16^{-}$), intermediate ($CD14^{++}/CD16^{+}$), and nonclassical patrolling ($CD14^{-}/CD16^{++}$) subpopulations based on surface markers, our previous analysis based on scRNA-seq data indicates that these cells show continuous transcriptional characteristics and trajectory analysis is an appropriate approach to characterize them (Li et al. 2020). This dataset has strong batch effects (Supplemental Fig. S16). To reconstruct the trajectories of these cells, for each method, we first

denoised and/or batch corrected the gene expression matrix, which was then fed into Monocle 3 (Cao et al. 2019) to estimate the pseudotime of each cell.

Fig. 5A shows that CarDEC yields the best pseudotime analysis results with cells from the three batches well mixed, and a clear pseudotemporal path emerged. The batchwise density plots show that the three batches have similar pseudotime distributions, suggesting that CarDEC successfully removed batch effects. The plots of *FCGR3A* (known marker gene for nonclassical monocytes) and *S100A8* (known marker gene for classical monocytes) gene expression also showed expected patterns (Supplemental Fig. S17). There are two key points of evidence from these marker gene plots suggesting that CarDEC recovered biological signal. First, for each marker gene, the expression levels are virtually identical across batches for all pseudotime points, which indicates that batch effects were removed for each gene expression and pseudotime relationship. Also, *FCGR3A* gene expression decreases monotonically with pseudotime, while *S100A8* expression increases monotonically with pseudotime. This is exactly the kind of behavior we expect from these marker genes. Since *FCGR3A* and *S100A8* are markers for the nonclassical and classical monocytes, respectively, we expect a good pseudotime analysis to segment the monocytes from nonclassical to classical (or vice versa) and for *FCGR3A* and *S100A8* expressions to be monotonic functions of pseudotime with opposite trends. By denoising and batch correcting gene counts, CarDEC successfully mixed batches and recovered biological signal down to the individual marker gene level.

By contrast, no other methods were able to achieve CarDEC's success in improving pseudotime analysis. Scanorama (Fig. 5B), DCA (Fig. 5C), and MNN (Fig. 5E) all failed to mix batches in the UMAP embedding from Monocle 3. Although scVI (Fig. 5D) mixed batches well in the UMAP embedding, the pseudotime distributions showed substantial variation across batches. For Scanorama, DCA, and MNN, neither of the

marker genes show strong monotonic trends as a function of the pseudotime, and for each marker gene, the relationship between expression and pseudotime varied by batch. These issues suggest that Scanorama, DCA, and MNN confounded biological signal, and obscured signals from canonical markers of established subpopulations, *FCGR3A* and *S100A8*, as marker genes.

There are other approaches to using these denoising and batch correction methods for pseudotime analysis. For example, one can subset the denoised and batch corrected matrix to include only the HVGs and then feed this into Monocle 3 (Supplemental Fig. S18, Supplemental Fig. S20). Alternatively, one can use the embedding from CarDEC, Scanorama, DCA, or scVI as the reduced dimension space to build the Monocle 3 pseudotime graph (Supplemental Fig. S19, Supplemental Fig. S21). In both of these other cases, the conclusions are largely the same, CarDEC is the best method for improving pseudotime analysis.

Next, we examined whether the denoised and batch corrected gene expression values can help improve gene expression quality for biological discovery. We focused our analyses on 61 transcription factors (TFs) that were expressed in the monocyte data and also found to be differentially expressed among classical, intermediate, and nonclassifcal monocytes by Wong *et al*. (Wong et al. 2011). Among these 61 TFs, 23 were selected as HVGs and the remaining 38 were designated LVGs. Fig. 6A shows that the CarDEC denoised gene expression revealed a gradually decreasing trend from nonclassical to classical for TFs that are known to be highly expressed in nonclassical monocytes, e.g., *TCF7L2*, *POU2F2*, *CEBPA*, and *HSBP1*. We also observed expected gene expression increase from nonclassical to classical for TFs that are known to be highly expressed in classical monocytes, e.g., *NFE2*, *CEBPD*, *GAS7*, and *MBD2*. Notably, some of the TFs with these expected expression patterns were not selected as HVGs, suggesting that denoising and batch correction in CarDEC helped recover the true biological variations in both HVGs and

LVGs. By contract, when using raw UMI counts as input, the heatmap did not reveal any meaningful biological patterns even for those TFs that were selected as HVGs (Fig. 6B).

An important task in trajectory analysis is to identify genes whose expression values change over pseudotime and whether the expression patterns are different between conditions (e.g., healthy vs diseased) over pseudotime. Avoiding false positive results is critical as failure of doing so may lead to follow-up of a wrong signal. Since the three batches were obtained from the same subject, we do not expect to detect significant gene expression differences over pseudotime among them. To this end, we performed differential expression analysis and compared the distributions of gene expression changes over pseudotime across the three batches. We performed hypothesis tests using the '*gam*' function in R package *mgcv* and tested whether gene expression patterns for the three batches are significantly different over pseudotime. Fig. 6C shows the p-values from this differential expression analysis for each method. CarDEC is much more effective in removing batch effects than Scanorama, DCA, scVI, and MNN. For the 23 HVG TFs, the median $-\log_{10}$ p-value for CarDEC is 0.30, whereas the median $-\log_{10}$ p-values for Scanorama, DCA, scVI, and MNN are 18.37, 5.29, 25.90, and 28.52, respectively. For the 38 LVG TFs, the median $-\log_{10}$ p-value for CarDEC is increased to 3.70, but still much lower than the other methods (9.81 for Scanorama, 6.36 for DCA, 39.19 for scVI, and 9.07 for MNN). These results indicate that failure to correct for batch effects could lead to a severe inflation of false positive results. Fig. 6D shows four selected TFs, where the denoised and batch corrected gene expression for CarDEC agreed well among the three batches, further confirming the effectiveness of CarDEC in removing batch effects in the gene expression space.

CarDEC is scalable to large dataset

As the scale of scRNA-seq continues to grow, it becomes increasingly important for a method to be scalable to large datasets. To evaluate the scalability of CarDEC, we leveraged a dataset of 104,694 human fetal liver cells (Popescu et al. 2019). Since we are principally interested in the problem of denoising and batch correcting in the full gene expression space, we retained all 21,521 genes after initial filtering for this analysis. For CarDEC we benchmarked two variations: a version that provides only denoised/batch corrected expression in the Z-score space (CarDEC Z-score) and a version that provides denoised/batch corrected expression in the count space (CarDEC Count).

We evaluated the runtime needed to process 10%, 20%, 40%, 60%, 80%, and 100% of cells in the human fetal liver dataset for CarDEC, Scanorama, DCA, scVI, and MNN. All evaluations were done on a 2019 edition MacBook Pro with 2.4 GHz 8-Core Intel Core i9 CPU and 32 GB of memory. CarDEC, DCA, and scVI were all trained with early stopping to halt training upon convergence. The results are shown in Supplemental Fig. S22. Both versions of CarDEC as well as DCA scaled approximately linearly with the number of cells and all three of these methods finished the analysis in less than 3.5 hours. We were unable to train scVI on more than 60% of the cells, as the jupyter kernel crashed mid-training for 80% of the cells. scVI also took considerably longer to run. At 60% of the data, CarDEC Z-score, DCA, and CarDEC Count took about 40 minutes, 1 hour, and 1.5 hours, respectively. By contrast, scVI took about 3 hours and 45 minutes to analyze this data. Scanorama had serious scalability issues. Like scVI, we could not run Scanorama for more than 60% of the data. Among the data points that we do have, Scanorama's run time is clearly not $O(n_{cells})$, while CarDEC, DCA, and scVI scale roughly linearly due to the minibatch gradient descent algorithm that trains all of them (any non-monotonicity is due to early stopping variation). Scanorama's run time appears to be parabolic as a function of sample size. MNN, also has serious scalability issues, which is consistent with a recent benchmarking study (Haghverdi et al. 2018). It took over 12 hours to analyze 20% of the data, and over 47 hours to analyze 40% of the data. We could not run MNN in under 48 hours using more than 40% of the data.

CarDEC is robust to hyperparameters

CarDEC involves several parameters that need to be tuned. To evaluate if CarDEC is robust to different choices of these parameters, we conducted additional analyses. First, we evaluated the performance of CarDEC for the α weight parameter, which is used when combining the self-supervised clustering loss and the reconstruction losses. The α parameter can be set to any value in the range of [0, 2]. Since a weight of 1 is the midpoint of this range, we chose it as a natural default. To evaluate the robustness of CarDEC to the choice of α, we varied its value across the set [0.1, 0.55, 1.0, 1.45, 1.9] and measured how the ARI changes for both the HVGs and the LVGs for each chosen value of α with all other parameters held equal. As shown in Supplemental Fig. S23, CarDEC is robust to this choice of parameter.

Next, we evaluated the performance of CarDEC when varying the number of clusters for clustering. To show that CarDEC is robust to this parameter, we cannot rely on ARI, since the maximum achievable ARI for any clustering method decreases the more the number of clusters is misspecified relative to the number of labels/cell types in the set of gold standard labels. Instead, we chose to show that increasing the number of clusters specified for CarDEC just splits existing clusters, without substantially changing cell type signals/separation. To do this, we fit CarDEC on the pancreas dataset repeatedly, varying the number of clusters with each fit. The numbers of clusters selected were 5, 9, 11, 14, 18, respectively. As shown in Supplemental Fig. 24A, even when the number of clusters is increased from 5 to 18, the cells were still separated mainly by their gold standard cell type labels. Closer examination of the UMAPs revealed that increasing the number of clusters usually just split cell types into two or more cluster label bins to accommodate surplus cluster labels, without changing the underlying structure of the UMAP plot. The CV score distributions shown in Supplemental Fig. 24B further supports that CarDEC is robust to the choice of the number of clusters since the CV score distributions were largely unchanged when

375 the number of clusters varied. We also visualized how cells in a cluster were split when CarDEC was refit
376 with an increasing number of clusters using a Sankey Plot. As shown in Supplemental Fig. 24C, in nearly
377 all cases, increasing the number of clusters just split existing clusters.
378

379 ## Discussion

380 We developed CarDEC, a joint deep learning model, that removes batch effects not only in the low-
381 dimensional embedding space but also across the entire gene expression space. As demonstrated in our
382 evaluations and a recent benchmarking study (Lucken et al. 2020), it is considerably harder to correct for
383 batch effects in the gene expression space than in the embedding space, and especially hard to correct
384 for batch effects in LVGs, which constitute the majority of the transcriptome. CarDEC was built to tackle
385 these challenges. To remove batch effects in the gene expression space, we minimize a loss function
386 that combines clustering and reconstruction losses. The self-supervised clustering loss, driven by HVGs,
387 regularizes the embedding and removes batch effects in the embedding. The rich, batch corrected
388 embedding is then used to compute an effectively batch corrected representation in the original gene
389 expression space. To address the difficulty associated with batch correcting LVGs, we implemented a
390 branching architecture, where embeddings are computed separately for HVGs and LVGs and where only
391 the HVG embedding is used to compute the clustering loss. Using the pancreatic islet datasets generated
392 from four scRNA-seq protocols, we demonstrated that this branching architecture substantially
393 improved batch effect removal on both the HVG and LVG gene expression spaces, as compared to the
394 naïve architecture.
395

396 Across a variety of datasets, with batch effects spanning multiple complexities in level and strength, we
397 demonstrated that CarDEC consistently led in its ability to remove batch effects. CarDEC was
398 consistently the best for removing batch effects in all capacities: in the embedding space, the HVG

399 expression space, and the LVG expression space. We showed that with appropriate denoising and batch
400 correction, the LVGs offer as much signal for clustering as the HVGs, suggesting that CarDEC has
401 substantially boosted the amount of information content in scRNA-seq. We also demonstrated that by
402 batch correcting gene expression counts, CarDEC improved pseudotemporal analysis of human
403 monocytes, an example of how batch correction can be used to improve downstream analyses.
404
405 CarDEC removes batch effects in the LVG denoised gene expression based on the embedding layer that
406 concatenates the HVG embedding. A potential concern of this concatenation is that the denoised LVG
407 expression values might contain artificially introduced signals from the HVGs. To examine this, we
408 focused on the bottom 16215 genes in the Pancreas dataset by variance and evaluated the performance
409 of CarDEC when varying the number of HVGs. Then, we fit CarDEC for various numbers of HVGs and
410 evaluated the ARI obtained from clustering the denoised expression for these 16215 genes. If the LVGs
411 contain artificial signals from the HVGs, then we would expect the ARI computed from the bottom
412 16125 genes to vary by how many HVGs were used to train CarDEC. As demonstrated in Supplemental
413 Fig. S25, this is not the case. We observed that ARI on the LVG set increased modestly when the number
414 of HVGs was increased from 500 to 1000, likely because 500 HVGs were too few highly variable features.
415 From 1000 up to 3000 inclusive, the ARI on the LVG set was very stable. After 3000 HVGs, there was a
416 modest drop in the LVG set's ARI, but this is likely because too many noisy features were introduced into
417 the HVG set, which hurt the quality of the KL-Divergence gradients. Since the 16125 LVG set's ARI is
418 fairly invariant to the number of HVGs used, especially in the range of 1000 to 3000 HVGs, we can infer
419 that the model architecture is not adding significant artificial signals from the HVGs to the LVGs.
420
421 In this paper, we focused on the analyses when using all genes as input. Since Scanorama, DCA, scVI, and
422 MNN are intended to be used with HVGs only, we reanalyzed every dataset with only HVGs selected as

the input using every method. As expected, many methods performed better when only HVGs were modeled (Supplemental Fig. S2, Supplemental Fig. S8, Supplemental Fig. S11, Supplemental Fig. S15). However, for the macaque retina data, Scanorama still struggled to remove its batch effects even when only HVGs were considered. For this challenging dataset, the ARI from Scanorama is only 0.18, which is much lower than all of the other methods (Supplemental Fig. S8).

Current scRNA-seq studies often include a large number of cells generated from many samples, across multiple conditions, and possibly using different protocols. Removing batch effects is critical for data integration. Since CarDEC provides efficient batch correction in the full gene expression space, it can be used for a wide array of analyses to facilitate biological discovery. Harmonized counts in the gene expression space can be used to estimate unbiased, batch corrected log fold changes, which can be used to identify marker genes for different cell types. These counts can also be used to reconstruct trajectories and identify genes showing pseudotemporal patterns. Lastly, CarDEC is computationally fast and memory efficient, making it a desirable tool for analyses of complex data in large-scale single-cell transcriptomics studies.

Figure legends

Figure 1. The workflow of CarDEC. The CarDEC workflow can be summarized in four steps that are depicted here: preprocessing, pretraining, denoising, and optionally, denoising on the count scale.

Figure 2. Justification for the branching architecture in CarDEC. The CarDEC API splits the input matrix into HVGs and LVGs and treats them separately with a "Branching" architecture as in Fig. 1. Alternatively, we can use a "Naïve" model that treats all features the same regardless of gene expression variance. The naïve model consists of an autoencoder with a clustering loss in addition to the reconstruction loss. Here we demonstrate the utility of the Branching architecture. The HVGs and LVGs were clustered separately, and the ARI of assignments is provided along with a UMAP plot. First row was colored by cell type, second by scRNA-seq protocol. (A) UMAP embedding computed from the denoised HVG counts for each method: CarDEC with Branching architecture, CarDEC with Naïve Architecture, Scanorama, scVI, and MNN. (B) UMAP embedding computed from the denoised LVG counts for each method. Figure legends are the same as those in (A). (C) Density plot of genewise coefficient of variation (CV) among batch centroids. Density plots are provided for HVGs and LVGs separately. (D) Clustering accuracy metrics were obtained using the embedding based methods to cluster the data, rather than running Louvain on the full gene expression space. Results for "Raw" were obtained by using Louvain's algorithm on the original HVG counts and are provided as a baseline with which to compare embedding based clustering results to.

Figure 3. Comparison of different methods on the macaque retina dataset. (A) UMAP embedding computed from the denoised HVG counts for each method. Top row was colored by cell type; bottom was colored by Macaque ID. Cells were also clustered with Louvain's algorithm. (B) UMAP embedding computed from the denoised LVG counts for each method. Figure legends are the same as those in (A).

(C) Density plot of genewise coefficient of variation (CV) among batch centroids. Density plots are provided for HVGs and LVGs separately. Centroids computed with sample id as batch definition. (D) Clustering accuracy metrics obtained using the embedding based methods to cluster the data, rather than running Louvain on the full gene expression space. Results for "Raw" were obtained by using Louvain's algorithm on the original HVG counts and are provided as a baseline with which to compare embedding based clustering results to.

Figure 4. Comparison of different methods on the mouse cortex dataset. (A) UMAP embedding computed from the denoised HVG counts for each method. Top row was colored by cell type; bottom was colored by batch. Cells were also clustered with Louvain's algorithm, and resultant ARI is provided. (B) UMAP embedding computed from the denoised LVG counts for each method. Figure legends are the same as those in (A). (C) Density plot of genewise coefficient of variation (CV) among batch centroids. Density plots are provided for HVGs and LVGs separately. (D) Clustering accuracy metrics obtained using the embedding-based methods to cluster the data, rather than running Louvain on the full gene expression space. Results for "Raw" were obtained by using Louvain's algorithm on the original HVG counts and are provided as a baseline with which to compare embedding based clustering results to.

Figure 5. Comparison of different methods for pseudotime analysis in the human monocyte data. The analysis is done on monocytes derived from three technical replicates from the same subject. For each method, the full dataset was denoised/batch corrected and then fed to Monocle 3 for pseudotime analysis. We show the UMAP embedding colored by batch (column 1) and estimated pseudotime (column 2). We also visualize the kernel density distribution of pseudotime by batch (column 3) and plot the distributions of marker genes *FCGR3A* and *S100A8* against pseudotime (columns 4 and 5, respectively). (A) Pseudotime analysis when using denoised/batch corrected gene expression matrix

from CarDEC as input. (B) Pseudotime analysis when using batch corrected gene expression matrix from Scanorama as input. (C) Pseudotime analysis when using denoised gene expression matrix from DCA but with Combat post-hoc batch correction as input. (D) Pseudotime analysis when using denoised/batch corrected gene expression matrix from scVI as input. (E) Pseudotime analysis when using batch corrected gene expression matrix from MNN as input.

Figure 6. Comparison of different methods for differential expression analysis of transcription factors in the human monocyte data. (A) Heatmap of scaled gene expression for CarDEC. Pseudotime was inferred based on embedding obtained from CarDEC using Monocle 3. (B) Heatmap of scaled raw UMI counts. Pseudotime was inferred based on embedding obtained from the scaled raw UMI counts using Monocle 3. (C) p-values obtained from differential expression analysis among the three batches over pseudotime. For each method, the pseudotime was inferred based on embedding obtained from the corresponding method. The red dotted line corresponds to p-value = 0.01. The top panel is for the 23 HVG TFs and the bottom panel is for the 38 LVG TFs. (D) Denoised and batch corrected gene expression for CarDEC, batch corrected gene expression for Scanorama, denoised gene expression for DCA with Combat post-hoc batch correction, denoised and batch corrected gene expression for scVI, and batch corrected gene expression for MNN over pseudotime for four selected TFs, *HIF1A*, *HES4*, *HSBP1*, and *GAS7*. For each method, the pseudotime was inferred based on embedding from the corresponding method.

Methods

The CarDEC workflow (Fig. 1) involves four steps: preprocessing, pretraining, gene expression denoising in Z-score space, and (optionally) denoising in count space. Below we briefly describe each of these steps. Details of the implementation is described in Supplemental Note 1, and the hyperparameters of CarDEC are shown in Supplemental Table 1.

Step 1: preprocessing

We first remove any cells expressing less than 200 genes, and then remove any genes expressed in less than 30 of the remaining cells. Let $\mathbf{X}$ be an $n \times p$ gene count matrix with n cells and p genes after filtering. The gene expression values are normalized. In the first step, cell level normalization is performed in which gene expression for a given gene in each cell is divided by the total gene expression across all genes in the cell, multiplied by the median total expression across all cells, and then transformed to a natural log scale. In the second step, gene level normalization is performed in which the cell level normalized values for each gene are standardized by subtracting the mean and dividing by the standard deviation across all cells within the same batch for the given gene. Highly variable genes (HVGs) are selected based on the log-normalized counts using the approach introduced by Stuart and Butler (Stuart et al. 2019a) and implemented in the "pp.highly_variable_genes" function with "batch_key" parameter in the SCANPY package (version >=1.4) (Wolf et al. 2018). The remaining genes that are not selected as HVGs are considered lowly variable genes (LVGs). We note that many of the LVGs still show cell-to-cell variability and are useful for clustering analysis after appropriate denoising and batch effect correction. We select 2,000 HVGs for all analyses in this paper.

Step 2: pretraining using the HVGs

The pretraining step is a straightforward implementation of an autoencoder. Let p_{HVG} be the number of HVGs selected in Step 1, and $\mathbf{Y}_{HVG}$ be the corresponding $n \times p_{HVG}$ matrix of normalized expression, subsetted to include only the HVGs. Define a standard autoencoder for $\mathbf{Y}_{HVG}$ with encoder and decoder represented by $f_{E,HVG}(\cdot\,;\, W_{E,HVG})$ and $f_{D,HVG}(\cdot\,;\, W_{D,HVG})$, respectively. The weights $W_{E,HVG}$ and $W_{D,HVG}$ are randomly initialized using the glorot uniform approach and are tuned during pretraining. We use the tanh activation for the output of the encoder, and the linear activation function for the output of the decoder. For all intermediate hidden layers in the encoder and decoder, we use the ReLu activation function. The autoencoder is pretrained with mean squared error loss using minibatch gradient descent with the Adam optimizer(Kingma and Ba 2015).

Step 3: denoising Z-scores

In this step, we use an expanded, branching architecture to accommodate LVGs, and introduce a clustering loss that regularizes the embedding and improves batch mixing and denoising especially in the gene space. Let p_{LVG} be the number of LVGs selected in Step 1, and $\mathbf{Y}_{LVG}$ be the corresponding $n \times p_{LVG}$ matrix of normalized expression, subsetted to include only the LVGs, and $\boldsymbol{y}_{i,HVG}$ and $\boldsymbol{y}_{i,LVG}$ be the vectors of HVGs and LVGs, respectively in cell i. We retain the encoder and decoder mappings for HVGs, $f_{E,HVG}(\cdot\,;\, W_{E,HVG})$ and $f_{D,HVG}(\cdot\,;\, W_{D,HVG})$ from Step 2, including the learnt weights $W_{E,HVG}$ and $W_{D,HVG}$. We introduce a clustering layer that takes the HVG embedding $\boldsymbol{z}_{i,HVG} = f_{E,HVG}(\boldsymbol{y}_{i,HVG}\,;\, W_{E,HVG})$ as input and returns for each cell a vector of cluster membership probabilities for h clusters, where h is a user-specified number. For this clustering layer, we introduce an $h \times d$ matrix of trainable weights/cluster centroids $\mathbf{M}$, where the j^{th} row of $\mathbf{M}$ is a cluster centroid $\boldsymbol{\mu}_j$, and d is dimension of the embedding.

To initialize $\mathbf{M}$, we run Louvain's algorithm on the embeddings $\{\boldsymbol{z}_{i,HVG}: i \in \{1, 2, \ldots, n\}\}$ learnt from the pretrained autoencoder and find the cluster centroid for each cluster. The clustering layer computes a vector of cluster membership probabilities for cell i, denoted by $\boldsymbol{q}_i$. Let q_{ij}, the j^{th} element of $\boldsymbol{q}_i$, denote the probability that cell i belongs to cluster j. Then the membership probabilities are computed using a t-distribution kernel as follows,

$$q_{ij} = \frac{\left(1 + \left\|\boldsymbol{z}_{i,HVG} - \mu_j\right\|^2\right)^{-1}}{\sum_{j'}\left(1 + \left\|\boldsymbol{z}_{i,HVG} - \mu_{j'}\right\|^2\right)^{-1}}$$

Since we do not have cell type labels in an unsupervised analysis, we create "pseudo-labels" that can be used in place of real labels for optimizing clustering weights. Inspired by Xie *et al.*(Xie et al. 2016), these pseudo-labels are computed from the membership probabilities q_{ij} as follows,

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'} q_{ij'}^2 / \sum_i q_{ij'}}$$

Let $\boldsymbol{p}_i$ be an h-dimensional vector whose j^{th} element is p_{ij}. Then the clustering loss for cell i is defined as the following Kullback–Leibler divergence (KLD),

$$l_{i,c} = KLD(\boldsymbol{p}_i || \boldsymbol{q}_i) = \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

This loss is a component of the total loss defined later. Since it takes the embedding vectors $\boldsymbol{z}_{i,HVG}$ as input, minimizing this objective function can refine the embedding and help to remove batch effects

from denoised counts computed using this embedding as input. The use of this KLD loss function was inspired by DESC(Li et al. 2020), which has shown that batch effects can be gradually removed over iterations. The intuition is to use "easy-to-cluster" cells, i.e., cells with "pseudo-labels", to guide the neural network to learn cluster-specific gene expression features while ignoring other unwanted noises such as batch effects. Over iterations, the algorithm ignores information unrelated to cell clustering by adjusting the network weights using the gradient descent algorithm and learns cluster-specific information. During the iterative update procedure, cells that are initially assigned to the same cluster are moved closer and closer to the cluster centroid, hence removing batch effects in the embedding space. Since the algorithm learns information on cell clusters from those "easy-to-cluster cells" while ignoring other irrelevant information by constructing the auxiliary distribution ***P*** and optimizing the KL-divergence between ***P*** and ***Q***, then as long as technical differences (e.g., between batches) are smaller than biological differences (e.g., between cell types), it can remove batch effect successfully.

We also introduce encoder and decoder mappings $f_{E,LVG}(\cdot\,;\,W_{E,LVG})$ and $f_{D,LVG}(\cdot\,;\,W_{D,LVG})$ to address the problem of denoising and batch correction for the LVGs. Unlike the HVG decoder $f_{D,HVG}$, the LVG decoder $f_{D,LVG}(\cdot\,;\,W_{D,LVG})$ does not map the low-dimension embedding $\boldsymbol{z}_{i,LVG}$ alone to reconstruct $\hat{\boldsymbol{y}}_{i,LVG}$ in the original p_{LVG}-dimension space. Rather, we concatenate the HVG and LVG embeddings together, and feed the combined vector $[\boldsymbol{z}_{i,HVG}\ \ \boldsymbol{z}_{i,LVG}]$ into the decoder to denoise and batch correct LVG expression in the original p_{LVG}-dimension space. That is,

$$\hat{\boldsymbol{y}}_{i,LVG} = f_{D,LVG}([\boldsymbol{z}_{i,HVG}\ \ \boldsymbol{z}_{i,LVG}];\ W_{D,LVG}).$$

This concatenated embedding is critical because it allows CarDEC to only use the high signal-to-noise ratio HVGs to drive the clustering loss, while still using the rich, batch corrected embedding that is

refined using this clustering loss to denoise and batch correct LVGs. The activation functions for the encoder and decoder of the LVGs are similarly defined as the autoencoder in Step 2.

To train this branching model, we first introduce two reconstruction losses, one for the HVGs and one for the LVGs computed as follows for cell i,

$$l_{i,HVG} = \frac{1}{p_{HVG}} \left\| \hat{\boldsymbol{y}}_{i,HVG} - \boldsymbol{y}_{i,HVG} \right\|^2$$

$$l_{i,LVG} = \frac{1}{p_{LVG}} \left\| \hat{\boldsymbol{y}}_{i,LVG} - \boldsymbol{y}_{i,LVG} \right\|^2$$

Then the total loss is calculated as a multi-component loss function as follows,

$$l_i = \alpha l_{i,c} + (2 - \alpha) \frac{l_{i,HVG} + l_{i,LVG}}{2},$$

where α is a hyperparameter ranging from 0 to 2 that balances reconstruction loss with clustering loss. We set α at 1 as default value. The total loss is minimized in an iterative fashion until certain convergence criteria are satisfied.

Step 4: denoising gene expression counts

In Step 3, the denoised expression values obtained from the decoder are on a Z-score scale and are not naturally comparable to raw UMI counts. To remedy this, we offer an optional downstream modeling step that provides denoised expression values on the original count scale. This strategy involves finding mean and dispersion parameters that maximize a negative binomial likelihood. We choose the negative

binomial distribution because previous studies have shown that UMI counts are not zero-inflated, and the negative binomial distribution fits the data well(Chen et al. 2018; Wang et al. 2018; Svensson 2020).

After the training in Step 3, we have obtained batch corrected low-dimension embeddings, $\boldsymbol{z}_{i,HVG}$ and $\boldsymbol{z}_{i,LVG}$ for each cell i from the fine-tuned HVG and LVG encoders. We will use two separate neural networks to maximize the negative binomial losses: one for the HVGs and one for the LVGs. These models are completely separate from one another but are trained almost identically with only minor differences. The goal is to map the embeddings into the full gene space to obtain mean and dispersion parameters for each gene. Without loss of generality, we use the HVGs as an example to illustrate how the neural network is built.

The vector of genewise means $\boldsymbol{\mu}_{i,HVG}$ and vector genewise dispersions $\boldsymbol{\theta}_{i,HVG}$ are given below,

$$\boldsymbol{\mu}_{i,HVG} = s_i \times exp\ (\mathbf{W}_{\mu,HVG} \times \tilde{\boldsymbol{z}}_{i,HVG}),$$

$$\boldsymbol{\theta}_{i,HVG} = softplu\, s\big(\mathbf{W}_{\theta,HVG} \times \tilde{\boldsymbol{z}}_{i,HVG}\big),$$

where s_i is the size factor for cell i, $\mathbf{W}_{\mu,HVG}$ and $\mathbf{W}_{\theta,HVG}$ are trainable weight matrices, and exp and $softplus$ are activation functions that are applied elementwise. For each gene j in cell i, we compute the negative log likelihood of the negative binomial distribution as

$$l_{ij} = -\log\left(\frac{\Gamma(x_{ij} + \theta_{ij})}{\Gamma(\theta_{ij})}\left(\frac{\theta_{ij}}{\theta_{ij} + \mu_{ij}}\right)^{\theta_{ij}}\left(\frac{\mu_{ij}}{\theta_{ij} + x_{ij}}\right)^{x_{ij}}\right),$$

where x_{ij} is the original count in HVG gene j for cell i, and where μ_{ij} and θ_{ij} are the j^{th} elements of $\boldsymbol{\mu}_{i,HVG}$ and $\boldsymbol{\theta}_{i,HVG}$, respectively. For the HVG count model, the full loss for cell i is then $l_i = \frac{1}{p_{HVG}} \sum_{j=1}^{p_{HVG}} l_{ij}$. The loss for the LVG count model can be similarly defined. Both the HVG and LVG count models are trained using their own early stopping and learning rate decay convergence monitoring.

Evaluation of batch effect removal in the gene expression space and the embedding space

Here, we briefly describe the workflow to evaluate batch effect removal and comparison between different methods (details see Supplemental Note 2). First, we evaluated the performance of different methods in removing batch effect in the gene expression space. For this evaluation, we considered CarDEC, Scanorama, DCA with post-hoc batch effect correction by Combat, scVI, and MNN. We ran all denoising/batch correction methods on the full data matrix and then present clustering results for denoised HVGs and denoised LVGs separately by subsetting the HVGs and LVGs from the full denoised/batch corrected expression matrix. The subsetted matrix that includes only the HVGs (or LVGs) is then passed down to the Louvain's clustering algorithm. All steps in this workflow are identical for both the HVGs and the LVGs, and all methods used the same HVGs and LVGs as input for clustering. Furthermore, on a given dataset, we benchmarked all methods with the same number of clusters. Second, we evaluated batch effect removal for the embedded representations of scRNA-seq. For this evaluation we considered CarDEC, Scanorama, DCA with post-hoc batch effect correction by Combat, scVI, and scDeepCluster. We excluded MNN since it has no embedding functionality. We included "raw" as a control method for comparison, which is just subsetting the raw data to include only the HVGs, and then running the clustering workflow.

Coefficient of variation (CV) analysis

To measure batch mixing, we examined the batchwise centroids before and after denoising. Let $\mathbf{X}'$ be a matrix of gene expression counts (including both HVGs and LVGs). $\mathbf{X}'$ can be the matrix of raw counts, or the denoised/batch corrected counts from any of CarDEC, Scanorama, DCA with post-hoc batch effect correction by Combat, scVI, or MNN. For CarDEC, we only considered denoised counts, not denoised expression in the Z-score space. If $\mathbf{X}'$ consists of MNN corrected expression, then we did not preprocess the data since MNN denoised expression is on a cosine scale. In the case of MNN, Scanorama and DCA after Combat correction, a fraction of expression counts can be negative, which poses difficulties when computing coefficients of variation. To circumvent this issue, any expression values after correction by these methods that are negative were truncated to zero for the CV analysis. For all other methods we have denoised expression in the non-negative count space, so we performed cell normalization and log normalization on $\mathbf{X}'$, exactly in the same way described in the Step 1 (preprocessing) of CarDEC.

Below we describe how the CV scores are calculated. Let ${x_{ij}}'$ be the expression value in gene j of cell i in $\mathbf{X}'$. Let S_{ab} be a set of integers defined such that $i \in S_{ab}$ if and only if cell i belongs to cell type a and is sequenced from batch b. Furthermore, let $c_{ajb} = \sum_{i \in S_{ab}} x_{ij} \,/ \sum_{i \in S_{ab}} 1$ be the centroid (mean expression) of batch b for gene j of cell type a. Let $C_{aj} = \{c_{ajb}\}$ be the collection of batch centroids for gene j in cell type a. We can now define a **cell-type-specific CV** as follows:

$$CV_{aj} = \frac{\sqrt{Var(C_{aj})}}{Mean(C_{aj}) + \gamma}$$

where $\gamma = 10^{-12}$ is a small constant to mitigate computational instability. A higher value of CV_{aj} corresponds to greater variation among batches and less batch mixing for cell type a, so a good batch effect removal method should drive CV_{aj} closer to zero. Normalizing by mean expression adjusts the

metric for how highly expressed the gene is, so that CVs from more highly expressed genes are comparable to CVs from less highly expressed genes.

To consolidate our results across cell types, so that we can present a single distribution of CV scores across all cell types, we combine the cell-type-specific CV scor$s$. Then we summarize the cell-type-specific CV scores by taking a weighted average of these scores across cell types, where the weight of a cell type is proportional to that cell type's frequency in the dataset. Specifically, the weight for cell type a is computed as $c(a) = \sum_i I(a_i = a) / \sum_i 1$, where a_i is the cell type of cell i. The combined CV score is calculated as follows:

$$CV_j = \sum_a c(a) CV_{aj}$$

Evaluation metrics for clustering

For all of our benchmark datasets, we used the cell type labels reported in the original papers as the gold standard. The clustering performance of each method was mainly evaluated using the adjusted rand index (ARI), calculated as below,

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}},$$

where n_{ij} is the number of cells in both cluster i from the cluster assignments obtained when benchmarking and in cell type j according to the gold standard cell type labels. a_i is the total number of cells in cluster i from the cluster assignments obtained when benchmarking, b_j is the total number of

695 cells in cell type j according to the gold standard cell type labels from the original study, and n is the
696 total number of cells. Additionally, we also computed normalized mutual information (NMI) and purity,
697 calculated as below,
698

$$NMI = 2 \times \frac{\sum_{ij} \frac{n_{ij}}{n} \log\left(\frac{n \times n_{ij}}{a_i \times b_j}\right)}{\sum_i \frac{a_i}{n} log\left(\frac{n}{a_i}\right) + \sum_j \frac{b_j}{n} log\left(\frac{n}{b_j}\right)},$$

699

$$Purity = \frac{1}{n} \sum_i \max_j n_{ij}.$$

700

701 Data sets

702 We analyzed multiple published scRNA-seq datasets, which are available through the accession numbers
703 reported in the original papers. 1) Human pancreatic islet data: CelSeq (Gene Expression Omnibus
704 GSE81076), CelSeq2 (Gene Expression Omnibus GSE85241), Fluidigm C1 (Gene Expression Omnibus
705 GSE86469), and Smart-seq2 (Array Express E-MTAB-5061); 2) Bipolar cells from mouse retina (Gene
706 Expression Omnibus GSE81904); 3) Bipolar cells from macaque retina (Gene Expression Omnibus
707 GSE118480); 4) mouse cortex data (Single Cell Portal SCP425); 5) human PBMC data (Single Cell Portal
708 SCP424); 6) human monocyte data GEO (GSE146974); 7) human fetal liver data (Array Express E-MTAB-
709 7407). Details of these datasets were described in Supplemental Table 2.
710

711 Software availability

712 An open-source implementation of the CarDEC algorithm can be downloaded from GitHub
713 (https://github.com/jlakkis/CarDEC) and is available as Supplemental Code. Code to reproduce all of our

714 analyses can be found on GitHub (https://github.com/jlakkis/CarDEC_Codes) and is available as
715 Supplemental Code as well.

716 Competing interest statement

717 The authors declare no competing interests.

718

719 Acknowledgments

720 This work was supported by the following grants: R01GM125301 (to M.L.), R01EY030192 (to M.L.),
721 R01EY031209 (to M.L.), R01HL113147 (to M.L. and M.P.R.), and R01HL150359 (to M.L. and M.P.R.). We
722 thank Sean Simmons and Jiarui Ding for their help on the mouse cortex and human PBMC data analysis.
723 We also thank Romain Lopez and Adam Gayoso for letting us know the scVI update, which has greatly
724 improved its performance. *Author contributions:* This study was conceived of and led by M.L.. J.L.
725 designed the model and algorithm, implemented the CarDEC software, and led data analysis with input
726 from M.L. and X.L.. X.L. led data analysis for the human monocyte data and designed the workflow
727 figure. D.W., G.H., and K.W. participated the early stage of algorithm design and testing. Y. Z. provided
728 input on memory management and analysis for the human monocyte data. L. U. provided input on the
729 model and algorithm design. H.Z. and M.P.R. provided input on the human monocyte data analysis. J.L.
730 and M.L. wrote the paper with feedback from all coauthors.

731

732 References

733 Barkas N, Petukhov V, Nikolaeva D, Lozinsky Y, Demharter S, Khodosevich K, Kharchenko PV. 2019. Joint
734 analysis of heterogeneous single-cell RNA-seq dataset collections. *Nat Methods* **16**: 695-698.
735 Cao J, Spielmann M, Qiu X, Huang X, Ibrahim DM, Hill AJ, Zhang F, Mundlos S, Christiansen L, Steemers FJ
736 et al. 2019. The single-cell transcriptional landscape of mammalian organogenesis. *Nature* **566**:
737 496-502.
738 Chen W, Li Y, Easton J, Finkelstein D, Wu G, Chen X. 2018. UMI-count modeling and differential
739 expression analysis for single-cell RNA sequencing. *Genome Biol* **19**: 70.

Ding J, Adiconis X, Simmons SK, Kowalczyk MS, Hession CC, Marjanovic ND, Hughes TK, Wadsworth MH, Burks T, Nguyen LT et al. 2020. Systematic comparison of single-cell and single-nucleus RNA-sequencing methods. *Nat Biotechnol* **38**: 737-746.

Eraslan G, Simon LM, Mircea M, Mueller NS, Theis FJ. 2019. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat Commun* **10**: 390.

Grun D, Muraro MJ, Boisset JC, Wiebrands K, Lyubimova A, Dharmadhikari G, van den Born M, van Es J, Jansen E, Clevers H et al. 2016. De Novo Prediction of Stem Cell Identity using Single-Cell Transcriptome Data. *Cell Stem Cell* **19**: 266-277.

Guo X, Gao L, Liu X, Yin J. 2017. Improved deep embedded clustering with local structure preservation. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*: 1753-1759.

Haghverdi L, Lun ATL, Morgan MD, Marioni JC. 2018. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat Biotechnol* **36**: 421-427.

Hicks SC, Townes FW, Teng M, Irizarry RA. 2018. Missing data and technical variability in single-cell RNA-sequencing experiments. *Biostatistics* **19**: 562-578.

Hie B, Bryson B, Berger B. 2019. Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. *Nature biotechnology* **37**: 685-691.

Johnson WE, Li C, Rabinovic A. 2007. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* **8**: 118-127.

Kingma DP, Ba JL. 2015. ADAM: a method for stochastic optimization. *International Conference on Learning Representation*.

Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, Baglaenko Y, Brenner M, Loh PR, Raychaudhuri S. 2019. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat Methods* **16**: 1289-1296.

Lahnemann D, Koster J, Szczurek E, McCarthy DJ, Hicks SC, Robinson MD, Vallejos CA, Campbell KR, Beerenwinkel N, Mahfouz A et al. 2020. Eleven grand challenges in single-cell data science. *Genome Biol* **21**: 31.

Lawlor N, George J, Bolisetty M, Kursawe R, Sun L, Sivakamasundari V, Kycia I, Robson P, Stitzel ML. 2017. Single-cell transcriptomes identify human islet cell signatures and reveal cell-type-specific expression changes in type 2 diabetes. *Genome Res* **27**: 208-222.

Li X, Wang K, Lyu Y, Pan H, Zhang J, Stambolian D, Susztak K, Reilly MP, Hu G, Li M. 2020. Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis. *Nat Commun* **11**: 2338.

Lopez R, Regier J, Cole MB, Jordan MI, Yosef N. 2018. Deep generative modeling for single-cell transcriptomics. *Nat Methods* **15**: 1053-1058.

Lucken MD, Buttner M, Chaichoompu K, Danese A, Interlandi M, Mueller MF, Strobl DC, Zappia L, Dugas M, Colome-Tatche M et al. 2020. Benchmarking atlas-level data integration in single-cell genomics. *bioRxiv* doi:https://doi.org/10.1101/2020.05.22.111161.

Muraro MJ, Dharmadhikari G, Grun D, Groen N, Dielen T, Jansen E, van Gurp L, Engelse MA, Carlotti F, de Koning EJ et al. 2016. A Single-Cell Transcriptome Atlas of the Human Pancreas. *Cell Syst* **3**: 385-394 e383.

Peng YR, Shekhar K, Yan W, Herrmann D, Sappington A, Bryman GS, van Zyl T, Do MTH, Regev A, Sanes JR. 2019. Molecular Classification and Comparative Taxonomics of Foveal and Peripheral Cells in Primate Retina. *Cell* **176**: 1222-1237 e1222.

Polanski K, Young MD, Miao Z, Meyer KB, Teichmann SA, Park JE. 2020. BBKNN: fast batch alignment of single cell transcriptomes. *Bioinformatics* **36**: 964-965.

Popescu DM, Botting RA, Stephenson E, Green K, Webb S, Jardine L, Calderbank EF, Polanski K, Goh I, Efremova M et al. 2019. Decoding human fetal liver haematopoiesis. *Nature* **574**: 365-371.

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Segerstolpe A, Palasantza A, Eliasson P, Andersson EM, Andreasson AC, Sun X, Picelli S, Sabirsh A, Clausen M, Bjursell MK et al. 2016. Single-Cell Transcriptome Profiling of Human Pancreatic Islets in Health and Type 2 Diabetes. *Cell Metab* **24**: 593-607.

Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck III WM, Hao Y, Stoeckius M, Smibert P, Satija R. 2019a. Comprehensive integration of single-cell data. *Cell* **177**: 1888-1902. e1821.

Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck WM, 3rd, Hao Y, Stoeckius M, Smibert P, Satija R. 2019b. Comprehensive Integration of Single-Cell Data. *Cell* **177**: 1888-1902 e1821.

Svensson V. 2020. Droplet scRNA-seq is not zero-inflated. *Nat Biotechnol* **38**: 147-150.

Tian T, Ji W, Qi S, Wei Z. 2019. Clustering single-cell RNA-seq data with a model-based deep learning approach. *Nature Machine Intelligence* **1**: 191-198.

Wang J, Huang M, Torre E, Dueck H, Shaffer S, Murray J, Raj A, Li M, Zhang NR. 2018. Gene expression distribution deconvolution in single-cell RNA sequencing. *Proc Natl Acad Sci U S A* **115**: E6437-E6446.

Welch JD, Kozareva V, Ferreira A, Vanderburg C, Martin C, Macosko EZ. 2019. Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity. *Cell* **177**: 1873-1887 e1817.

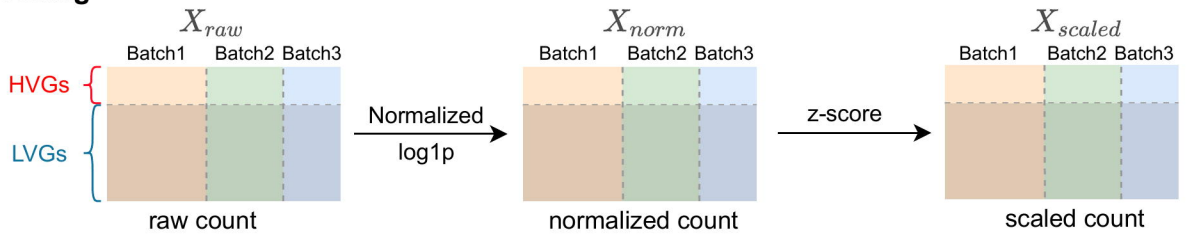
Wolf FA, Angerer P, Theis FJ. 2018. SCANPY: large-scale single-cell gene expression data analysis. *Genome biology* **19**: 15.

Wong KL, Tai JJ, Wong WC, Han H, Sem X, Yeap WH, Kourilsky P, Wong SC. 2011. Gene expression profiling reveals the defining features of the classical, intermediate, and nonclassical human monocyte subsets. *Blood* **118**: e16-31.

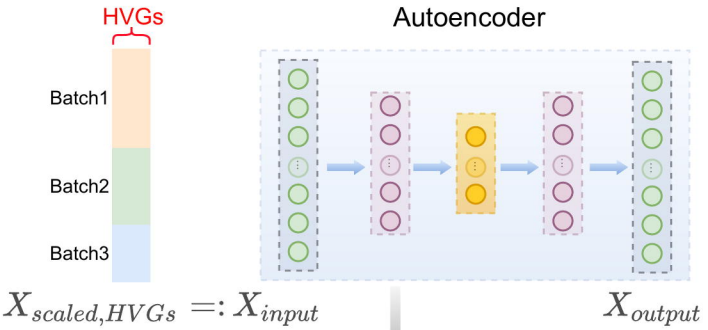
Xie J, Girshick R, Farhadi A. 2016. Unsupervised deep embedding for clustering analysis. *Proceedings of International Conference on Machine Learning*: 478-487.

Workflow of CarDEC

Step 1: Preprocessing



Step 2: Pretraining

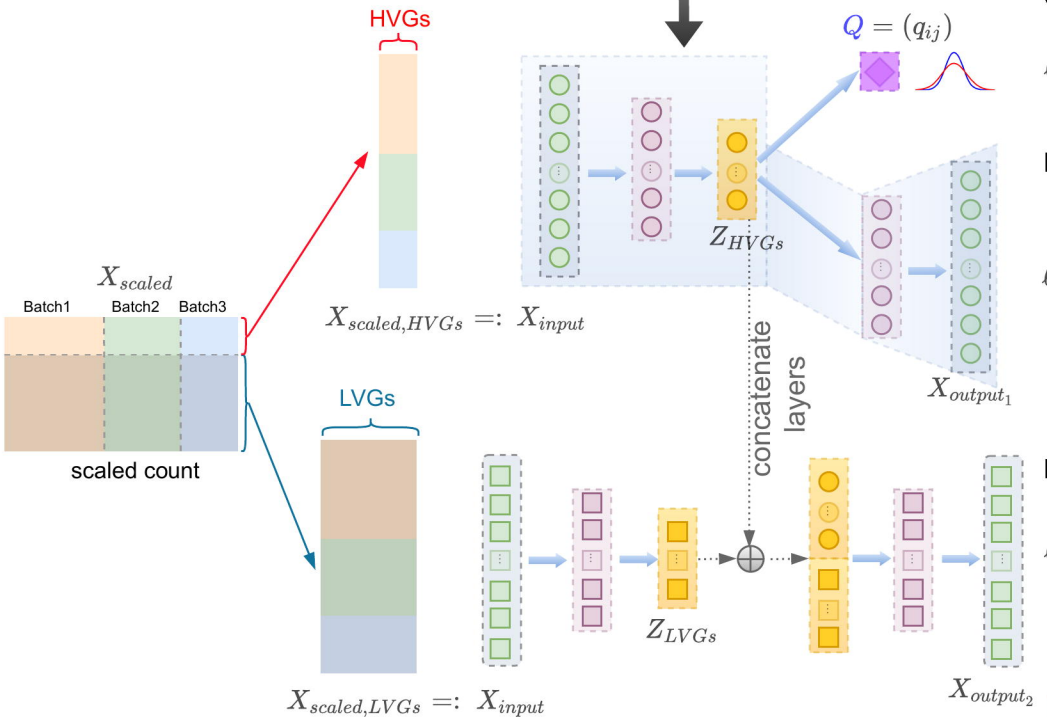


Reconstruction loss with only HVGs:

$$\ell = \frac{1}{p_{HVGs}} \| X_{scaled,HVGs} - X_{output} \|^2$$

Transfer encoder weights and decoder weights

Step 3: Denoised z-score



Clustering loss:

$$\ell_c = KL(P\|Q) = \sum_{i=1}^{n}\sum_{j=1}^{K} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Reconstruction loss with only HVGs:

$$\ell_{HVGs} = \frac{1}{p_{HVGs}} \| X_{scaled,HVGs} - X_{output_1} \|^2$$

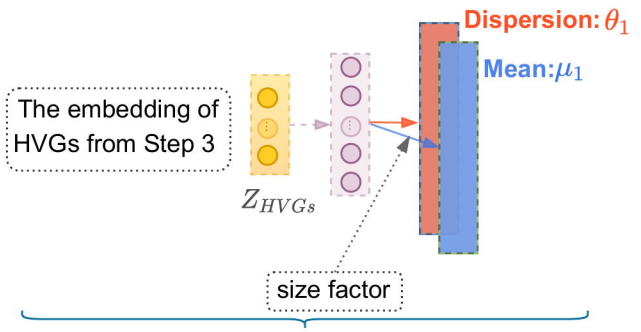
Reconstruction loss with only LVGs:

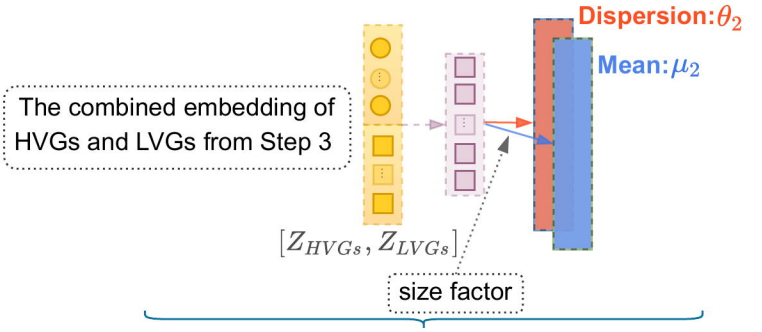
$$\ell_{LVGs} = \frac{1}{p_{LVGs}} \| X_{scaled,LVGs} - X_{output_2} \|^2$$

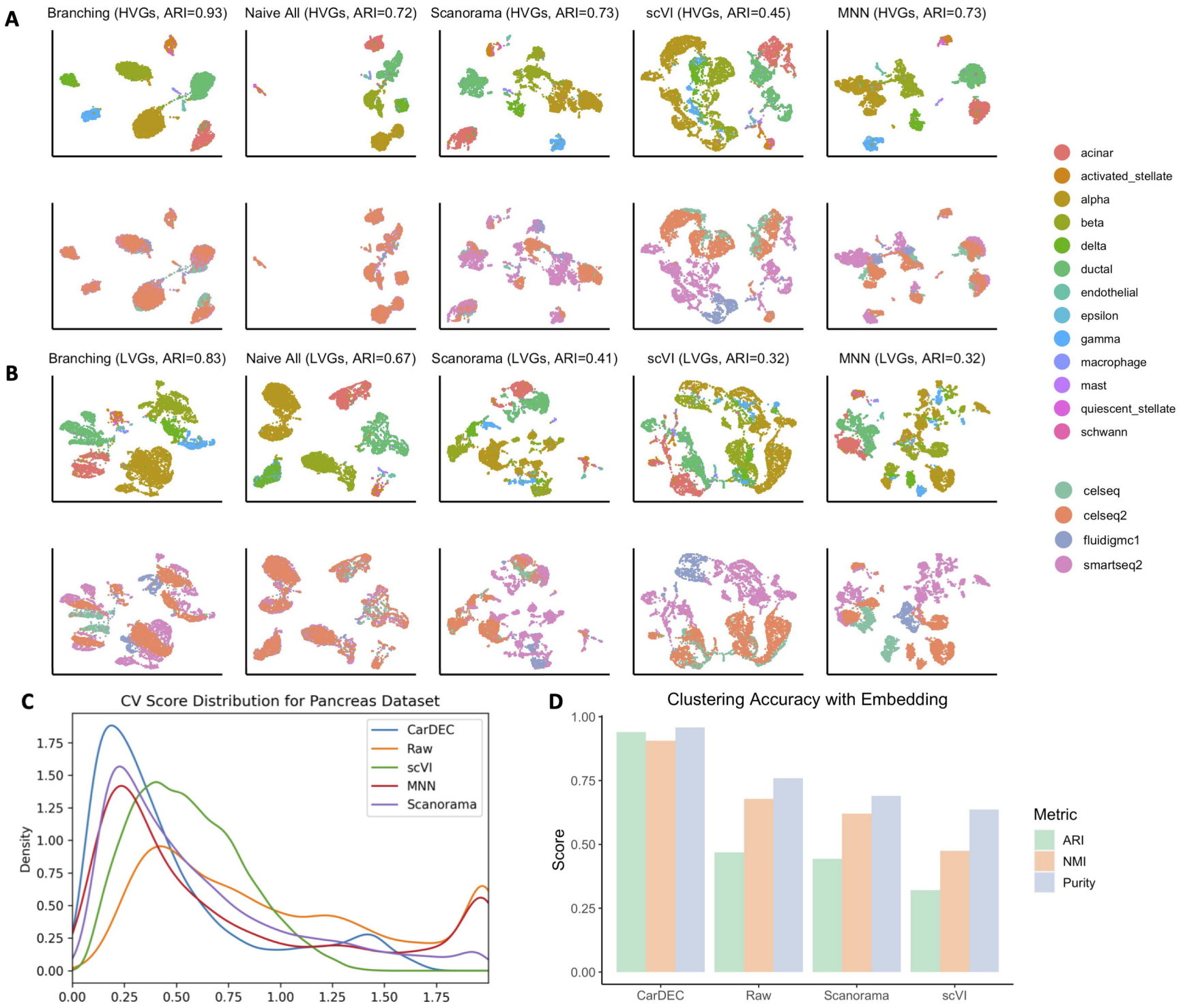
Total loss:

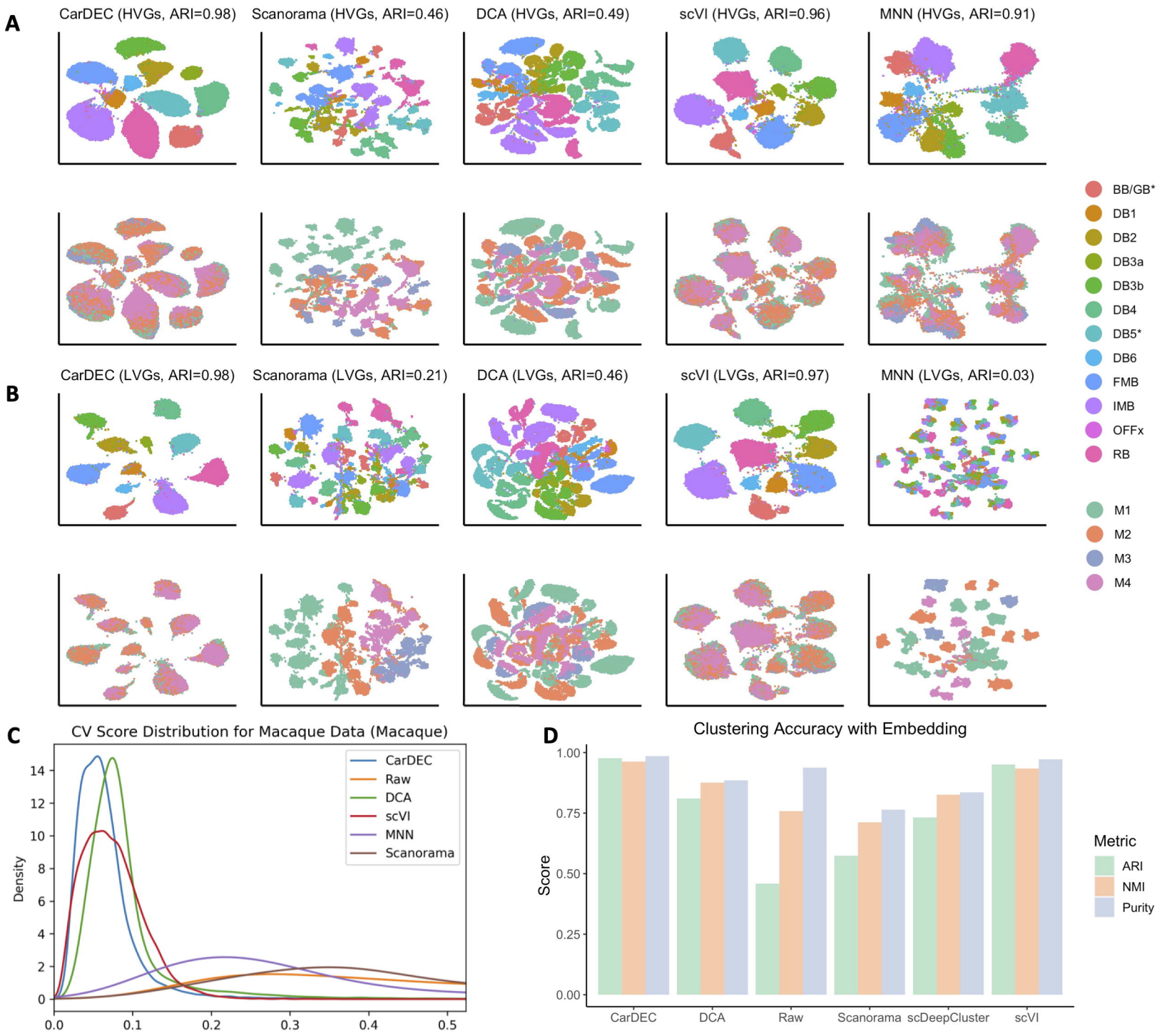
$$\ell_{total} = (2-\lambda)\ell_c + \lambda(\ell_{HVGs} + \ell_{LVGs})/2$$

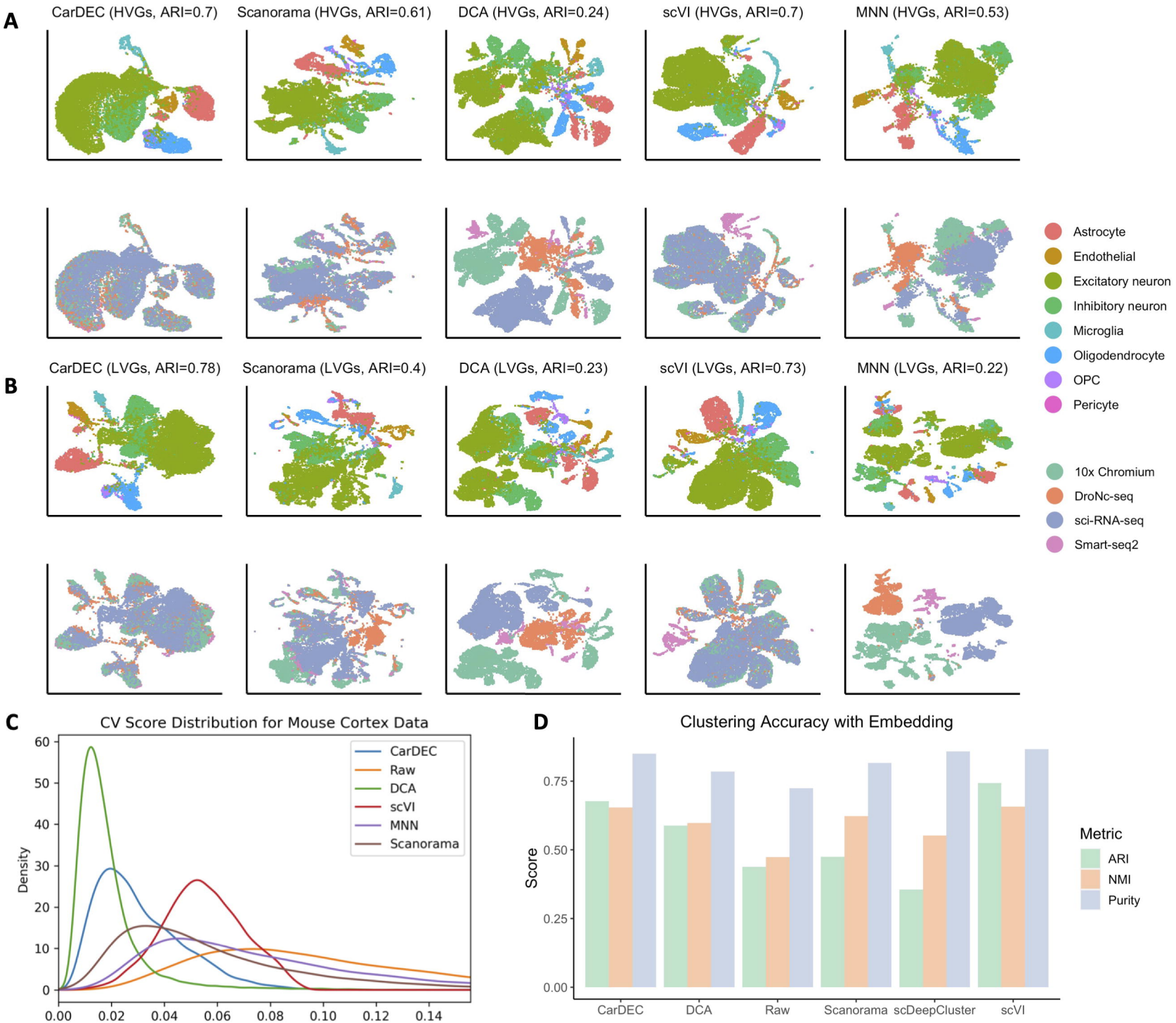
Step 4: Denoised raw count (optional)

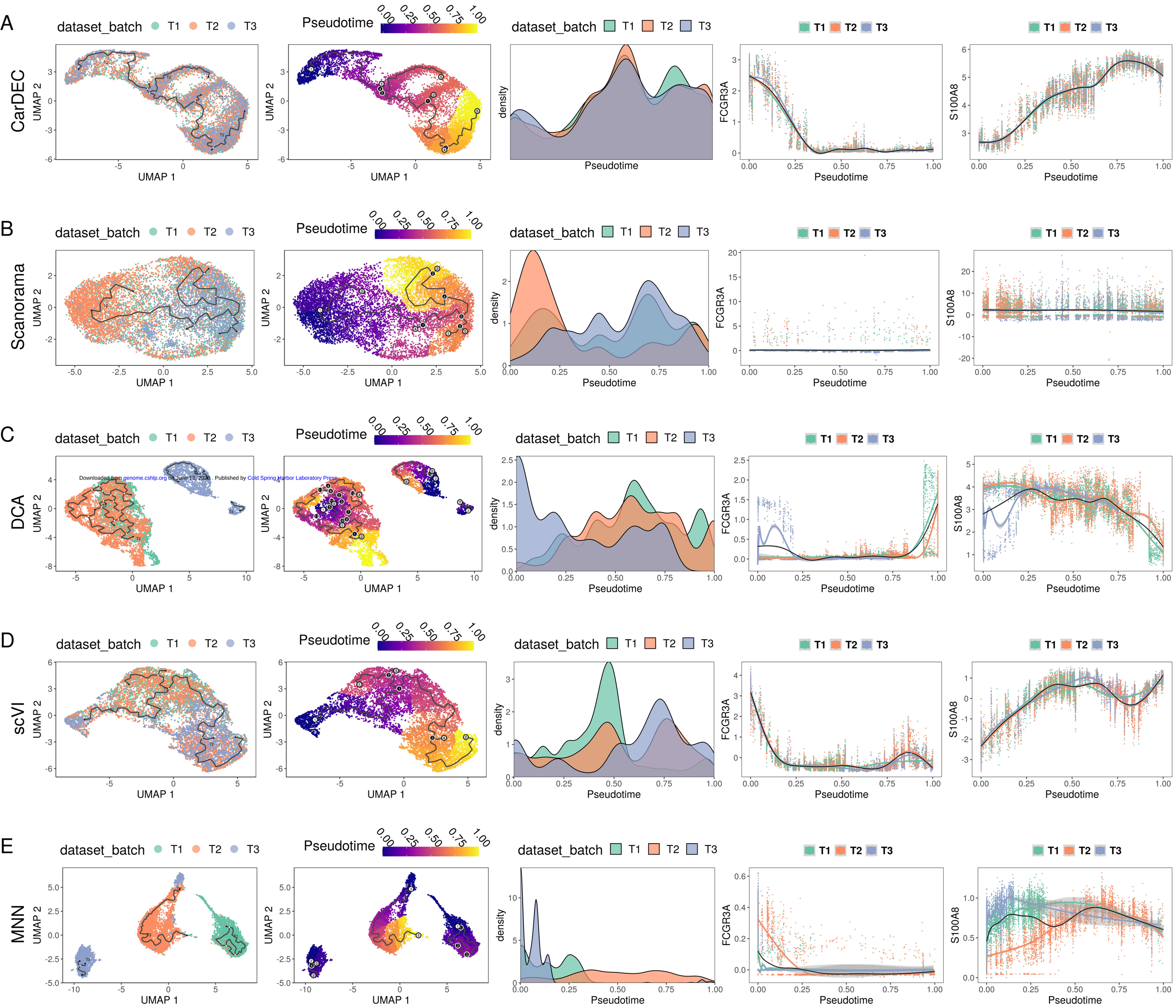


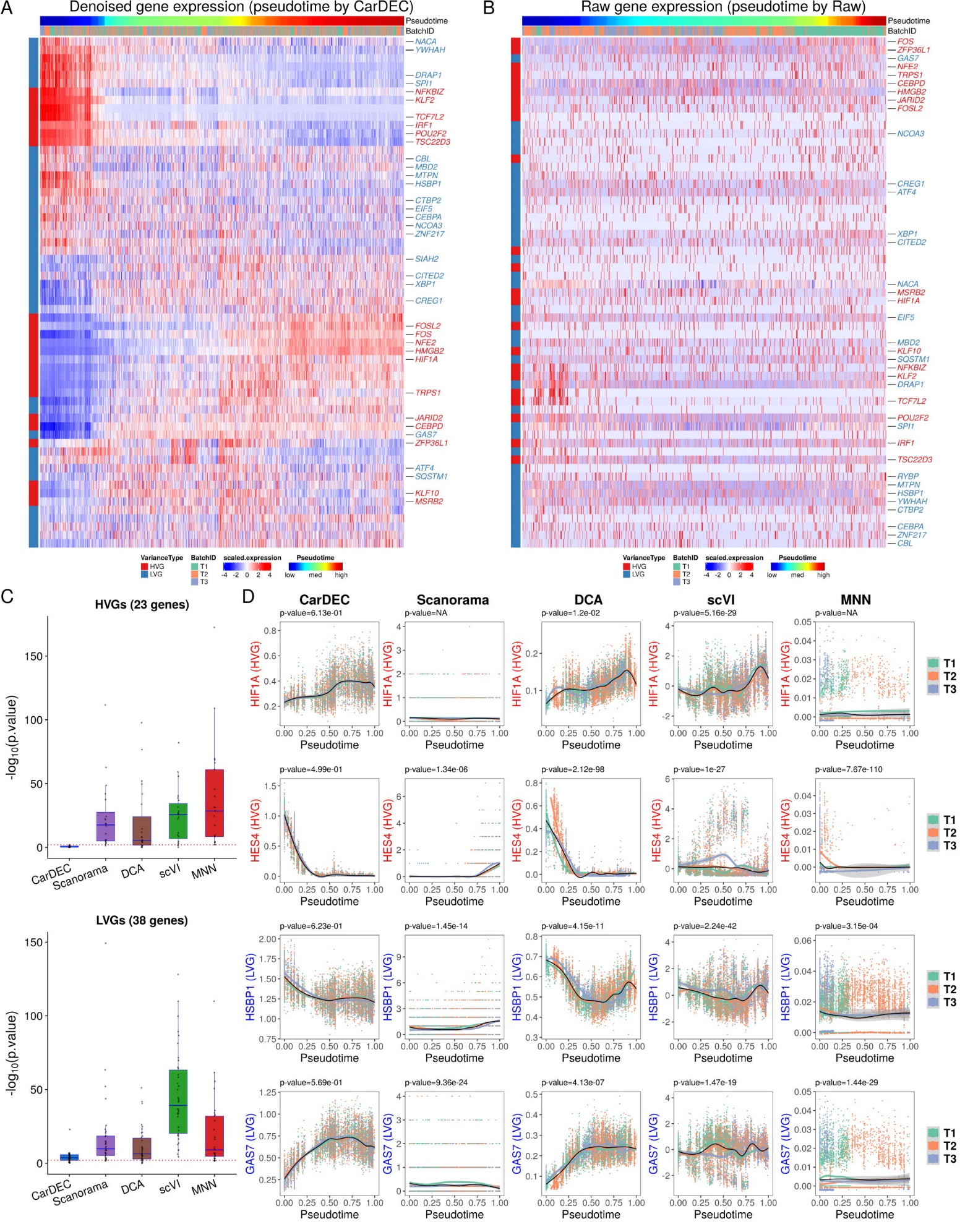



A
Branching (HVGs, ARI=0.93)
Naive All (HVGs, ARI=0.72)
Scanorama (HVGs, ARI=0.73)
scVI (HVGs, ARI=0.45)
MNN (HVGs, ARI=0.73)
B
Branching (LVGs, ARI=0.83)
Naive All (LVGs, ARI=0.67)
Scanorama (LVGs, ARI=0.41)
scVI (LVGs, ARI=0.32)
MNN (LVGs, ARI=0.32)
acinar
activated_stellate
alpha
beta
delta
ductal
endothelial
epsilon
gamma
macrophage
mast
quiescent_stellate
schwann
celseq
celseq2
fluidigmc1
smartseq2
C
CV Score Distribution for Pancreas Dataset
CarDEC
Raw
scVI
MNN
Scanorama
Density
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
D
Clustering Accuracy with Embedding
Score
0.00
0.25
0.50
0.75
1.00
CarDEC
Raw
Scanorama
scVI
Metric
ARI
NMI
Purity


A
CarDEC (HVGs, ARI=0.98)
Scanorama (HVGs, ARI=0.46)
DCA (HVGs, ARI=0.49)
scVI (HVGs, ARI=0.96)
MNN (HVGs, ARI=0.91)
B
CarDEC (LVGs, ARI=0.98)
Scanorama (LVGs, ARI=0.21)
DCA (LVGs, ARI=0.46)
scVI (LVGs, ARI=0.97)
MNN (LVGs, ARI=0.03)
BB/GB*
DB1
DB2
DB3a
DB3b
DB4
DB5*
DB6
FMB
IMB
OFFx
RB
M1
M2
M3
M4
C
CV Score Distribution for Macaque Data (Macaque)
Density
CarDEC
Raw
DCA
scVI
MNN
Scanorama
D
Clustering Accuracy with Embedding
Score
CarDEC
DCA
Raw
Scanorama
scDeepCluster
scVI
Metric
ARI
NMI
Purity


A
CarDEC (HVGs, ARI=0.7)
Scanorama (HVGs, ARI=0.61)
DCA (HVGs, ARI=0.24)
scVI (HVGs, ARI=0.7)
MNN (HVGs, ARI=0.53)
B
CarDEC (LVGs, ARI=0.78)
Scanorama (LVGs, ARI=0.4)
DCA (LVGs, ARI=0.23)
scVI (LVGs, ARI=0.73)
MNN (LVGs, ARI=0.22)
Astrocyte
Endothelial
Excitatory neuron
Inhibitory neuron
Microglia
Oligodendrocyte
OPC
Pericyte
10x Chromium
DroNc-seq
sci-RNA-seq
Smart-seq2
C
CV Score Distribution for Mouse Cortex Data
Density
CarDEC
Raw
DCA
scVI
MNN
Scanorama
D
Clustering Accuracy with Embedding
Score
CarDEC
DCA
Raw
Scanorama
scDeepCluster
scVI
Metric
ARI
NMI
Purity




A
Denoised gene expression (pseudotime by CarDEC)
Pseudotime
BatchID
NACA
YWHAH
DRAP1
SPI1
NFKBIZ
KLF2
TCF7L2
IRF1
POU2F2
TSC22D3
CBL
MBD2
MTPN
HSBP1
CTBP2
EIF5
CEBPA
NCOA3
ZNF217
SIAH2
CITED2
XBP1
CREG1
FOSL2
FOS
NFE2
HMGB2
HIF1A
TRPS1
JARID2
CEBPD
GAS7
ZFP36L1
ATF4
SQSTM1
KLF10
MSRB2
B
Raw gene expression (pseudotime by Raw)
Pseudotime
BatchID
FOS
ZFP36L1
GAS7
NFE2
TRPS1
CEBPD
HMGB2
JARID2
FOSL2
NCOA3
CREG1
ATF4
XBP1
CITED2
NACA
MSRB2
HIF1A
EIF5
MBD2
KLF10
SQSTM1
NFKBIZ
KLF2
DRAP1
TCF7L2
POU2F2
SPI1
IRF1
TSC22D3
RYBP
MTPN
HSBP1
YWHAH
CTBP2
CEBPA
ZNF217
CBL
VarianceType
HVG
LVG
BatchID
T1
T2
T3
scaled.expression
-4 -2 0 2 4
Pseudotime
low med high
C
HVGs (23 genes)
-log10(p.value)
CarDEC
Scanorama
DCA
scVI
MNN
LVGs (38 genes)
D
CarDEC
Scanorama
DCA
scVI
MNN
p-value=6.13e-01
p-value=NA
p-value=1.2e-02
p-value=5.16e-29
p-value=NA
HIF1A (HVG)
p-value=4.99e-01
p-value=1.34e-06
p-value=2.12e-98
p-value=1e-27
p-value=7.67e-110
HES4 (HVG)
p-value=6.23e-01
p-value=1.45e-14
p-value=4.15e-11
p-value=2.24e-42
p-value=3.15e-04
HSBP1 (LVG)
p-value=5.69e-01
p-value=9.36e-24
p-value=4.13e-07
p-value=1.47e-19
p-value=1.44e-29
GAS7 (LVG)
Pseudotime
T1
T2
T3