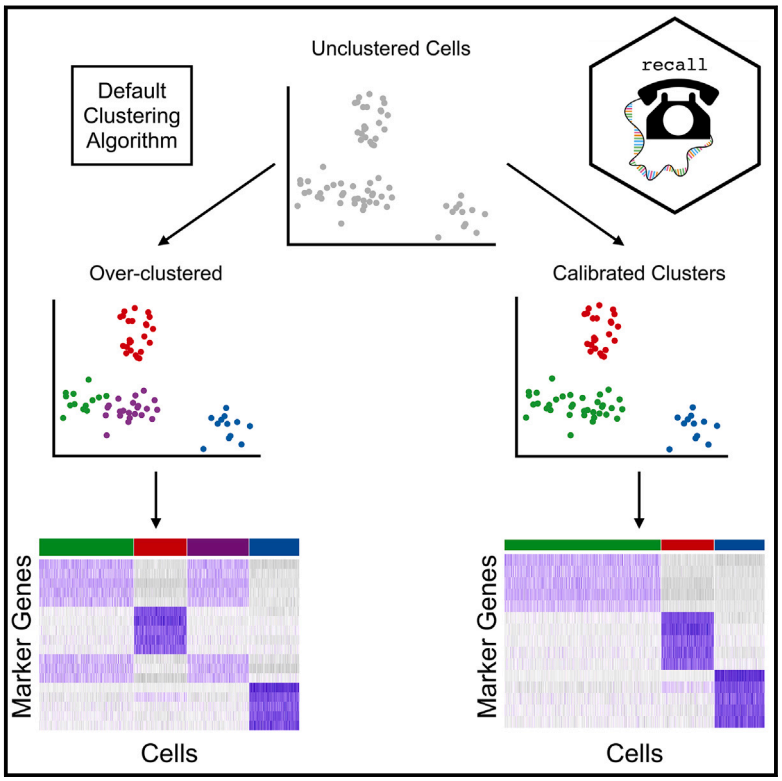
Artificial variables help to avoid over-clustering in single-cell RNA sequencing

Graphical abstract



Authors

Alan DenAdel, Michelle L. Ramseier, Andrew W. Navia, ..., Peter S. Winter, Ava P. Amini, Lorin Crawford

Correspondence

lcrawford@microsoft.com

Calibrated clustering with artificial variables protects against over-clustering single-cell RNA-seq data by controlling for the impact of reusing the same data twice when performing differential expression analysis, commonly known as "double dipping."

DenAdel et al., 2025, The American Journal of Human Genetics *112*, 940–951
April 3, 2025
https://doi.org/10.1016/j.ajhg.2025.02.014

CellPress

Artificial variables help to avoid over-clustering in single-cell RNA sequencing

Alan DenAdel,[1] Michelle L. Ramseier,[2,3,4,5,6] Andrew W. Navia,[3] Alex K. Shalek,[2,3,4,5,6] Srivatsan Raghavan,[3,7,8,9,11] Peter S. Winter,[3,11] Ava P. Amini,[10,11] and Lorin Crawford[10,11,*]

Summary

Standard single-cell RNA sequencing (scRNA-seq) pipelines nearly always include unsupervised clustering as a key step in identifying biologically distinct cell types. A follow-up step in these pipelines is to test for differential expression between the identified clusters. When algorithms over-cluster, downstream analyses can produce misleading results. In this work, we present "recall" (calibrated clustering with artificial variables), a method for protecting against over-clustering by controlling for the impact of reusing the same data twice when performing differential expression analysis, commonly known as "double dipping." Importantly, our approach can be applied to a wide range of clustering algorithms. Using real and simulated data, we show that recall provides state-of-the-art clustering performance and can rapidly analyze large-scale scRNA-seq studies, even on a personal laptop.

Introduction

Recent advances in single-cell RNA sequencing (scRNA-seq) technologies have enabled the generation of datasets that contain the transcriptomic profiles of thousands to millions of individual cells.[1,2] Unless an additional assay is paired with sequencing (e.g., cellular indexing of transcriptomes and epitopes, known as CITE-seq[3]), cell type labels are not provided with the corresponding transcriptomic profiles. This has led to many scRNA-seq bioinformatic pipelines requiring both (1) clustering to identify putative cell types based on shared gene expression covariation and (2) differential gene expression analysis between cells in each cluster to identify "marker genes" uniquely expressed by each putative cell type. The most commonly used software packages, such as Seurat[4] and Scanpy,[5] perform these two steps on the same dataset. This double use of data is often referred to as "circular analysis" or "double dipping" and is known to result in highly inflated p values, even in the null case when gene expression is identically distributed and there are no true groupings that distinguish cell populations.[6,7] The miscalibrated test statistics produced by circular analyses make it challenging to assess whether the genes found to be differentially expressed between two putative cell groups are "real" or solely identified due to chance based on the way cells are partitioned by the clustering algorithm being used. Importantly, simple solutions, such as sample splitting between cells, do not appropriately correct for this type of post-selective inference.[7]

Several methods have been recently developed to correct for post-selective inference after clustering. These methods include (1) an approximate test based on truncated normal distributions,[8] (2) a data splitting strategy that splits data at the level of individual gene counts,[7] and (3) using synthetic null variables called knockoffs to calibrate hypothesis testing.[6] The point of each of these methods is to identify an appropriate hypothesis testing significance threshold to account for the statistical inflation that occurs due to the double use of data. However, none of these tests inform if (or how) the re-clustering of cells should be done. They simply return a list of calibrated p values. As a result, approaches for protecting against over-clustering have recently been proposed, including "single-cell significance of hierarchical clustering" (sc-SHC),[9] "clustering hierarchy optimization by iterative random forests" (CHOIR),[10] and an "adaptive embedding and clustering method" using variational autoencoders (scAce).[11]

In this work, we take inspiration from the use of negative control knockoff variables for calibrated statistical tests[12,13] and introduce "recall," a method for performing calibrated clustering with artificial variables in single-cell datasets. The ultimate goal of recall is to provide users with the correct number of clusters. The rationale is that when clusters are correctly inferred, the effect of double dipping on downstream tasks (e.g., differential expression analyses) is minimal. Our approach can be paired with any existing clustering algorithm that has a hyperparameter for tuning the number of clusters and makes no strong assumptions about the input data. We statistically motivate the need for an algorithm like

[1]Center for Computational Molecular Biology, Brown University, Providence, RI 02912, USA; [2]Institute for Medical Engineering and Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; [3]Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA; [4]Koch Institute for Integrative Cancer Research, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; [5]Department of Chemistry, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; [6]Ragon Institute of MGH, MIT, and Harvard, Cambridge, MA 02139, USA; [7]Department of Medical Oncology, Dana-Farber Cancer Institute, Boston, MA 02215, USA; [8]Harvard Medical School, Boston, MA 02115, USA; [9]Department of Medicine, Brigham and Women's Hospital, Boston, MA 02115, USA; [10]Microsoft Research, Cambridge, MA 02142, USA
[11]These authors contributed equally
*Correspondence: lcrawford@microsoft.com
https://doi.org/10.1016/j.ajhg.2025.02.014.

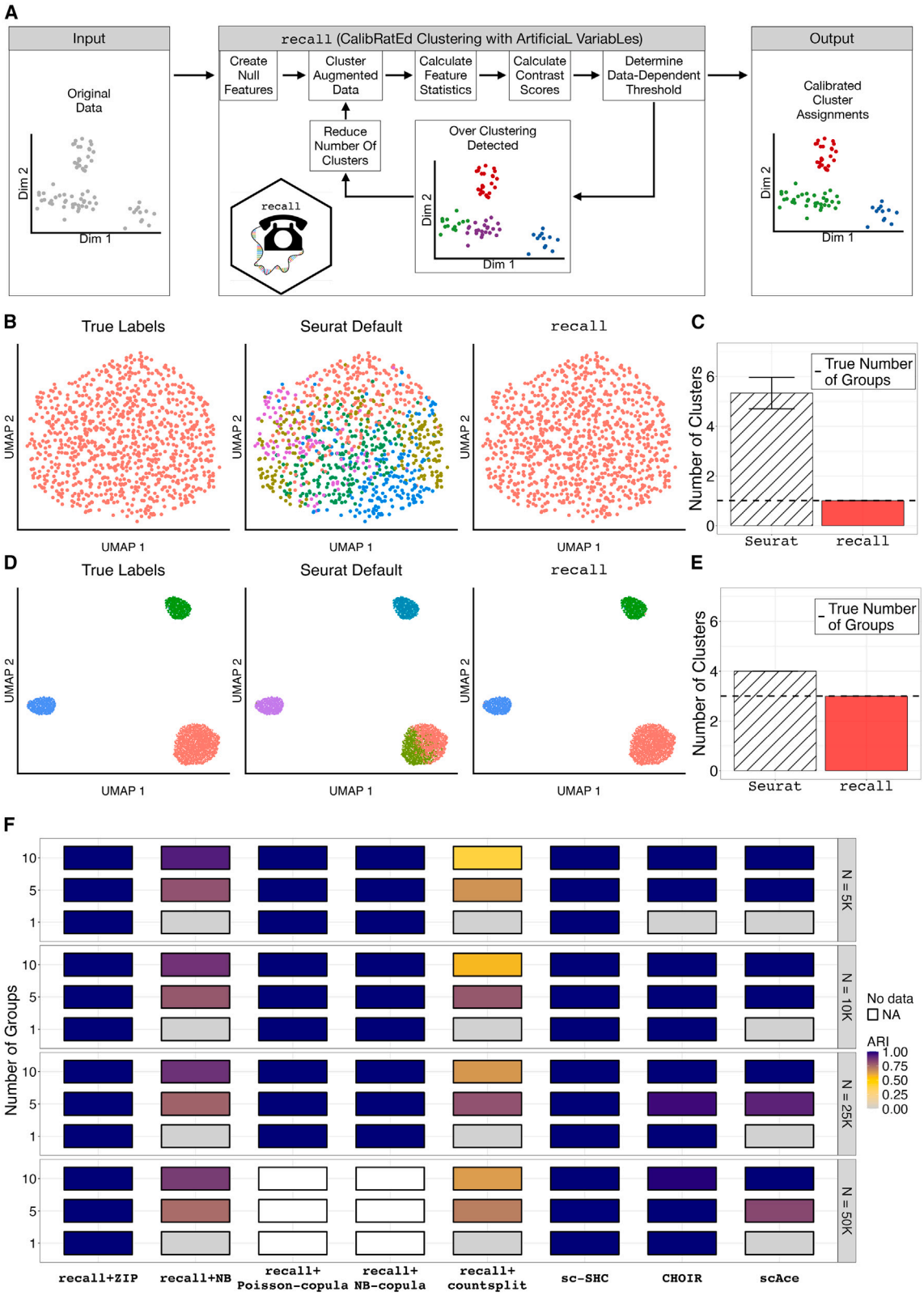


Figure 1. Overview of the recall algorithm and examples of results from different clustering approaches on simple simulated datasets

(A) Schematic of the clustering workflow with the recall approach.

(B) Demonstration of the traditional clustering framework versus the alternative using recall for simulated data with one known group. Images left to right show the true labels, clusters found using the Louvain algorithm with default parameter settings in Seurat, and the clusters found using the same Louvain algorithm paired with recall.

(legend continued on next page)

recall, evaluate its utility against other recently proposed clustering correction methods, and demonstrate its ability to efficiently scale to large-scale scRNA-seq studies.

Material and methods

Overview of the recall algorithm

Consider a study with scRNA-seq expression data for $i = 1, \ldots, N$ cells that each have measurements for $j = 1, \ldots, G$ genes. Let this dataset be represented by an $N \times G$ matrix $\mathbf{X}$ where the column-vector $\mathbf{x}_j$ denotes the expression profile for the j-th gene. The recall method augments the real expression matrix with artificial null genes, which are generated to have no association with any particular cell type.[12,13] These negative control variables go through the same preprocessing, clustering, and differential expression analyses as the real observed genes in the study; therefore, they are presented with the same opportunity to be identified as marker genes. Since the artificial null genes are essentially noise variables, the distribution of their test statistics represents the impact of post-selective inference (i.e., deviations from the null). As a result, we can correct for these same deviations from the null in the observed test statistics for the real genes, which allows us to also calibrate our cluster assignments. This process is akin to implementing a "knockoff filter" (which controls the false discovery rate) when testing for differentially expressed genes between clusters.[12,13] If there are no detectable differences between the inferred clusters, we assume that over-clustering has occurred and re-cluster with a smaller number of groups. More specifically, recall works by implementing the following steps (Figure 1A).

(1) For each gene in the study $\mathbf{x}_j$, generate an artificial null expression vector $\tilde{\mathbf{x}}_j$. Next, concatenate all of the synthetic genes together and construct a matrix of artificial null variables $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_G]$.
(2) Combine the real gene expression matrix with the artificial null features into a single object $\mathbf{X}^* = [\mathbf{X}; \tilde{\mathbf{X}}]$. Then, perform the usual preprocessing on the augmented data matrix $\mathbf{X}^*$. In this paper, preprocessing consists of normalizing the expression counts followed by principal-component analysis (PCA).
(3) Apply a given clustering algorithm (e.g., the Louvain algorithm) to the PCA embeddings of the augmented matrix $\mathbf{X}^*$ (or, alternatively, apply the clustering algorithm to the augmented matrix directly).
(4) Conduct differential expression analysis between each k-th and l-th cluster pair, denoted by C_k and C_l, respectively. Obtain p values for all genes (real and artificial nulls) across each comparison.
(5) Let $p_j(k;l)$ represent the p value for the j-th real gene when comparing differential expression between clusters C_k and C_l. Similarly, let $\tilde{p}_j(k;l)$ represent the p value for the same comparison but for the corresponding j-th artificial null gene. We use these two p values to compute the following test statistic (or contrast score)[6,12]:

$$W_j(k;l) = -\log p_j(k;l) - \left[-\log \tilde{p}_j(k;l)\right]. \qquad \text{(Equation 1)}$$

Intuitively, a large, positive value of $W_j(k;l)$ represents evidence that the j-th gene is truly different between clusters C_k and C_l, while a value less than or equal to zero represents strong evidence that there is no difference in the expression of the j-th gene between the groups. To compute the test statistics $W_j(k;l)$ for each cluster in Equation 1, recall uses p values $p_j(k;l)$ and $\tilde{p}_j(k;l)$ from the Wilcoxon rank-sum test as implemented by the FindMarkers function in the Seurat software package[4] and accelerated by Presto.[14]

(6) Next, compute the data-dependent threshold (inspired by the knockoff+ method proposed by Barber and Candès[12]) via the following formulation:

$$\tau(k,l) = \min\left\{t > 0 : \frac{1 + \#\{j : W_j(k;l) \leq -t\}}{\max\{\#\{j : W_j(k;l) \geq t\}, 1\}} \leq q\right\}, \qquad \text{(Equation 2)}$$

where $\#\{\cdot\}$ denotes the cardinality of a set and q is a hyperparameter (in the knockoff framework, q represents the desired false discovery rate). By default and for all results presented in this paper, recall sets $q = 0.05$. If no such $t > 0$ exists, we set $\tau(k,l) = \infty$.

If, for any pair of clusters, $\tau(k,l) = \infty$, then we return to step 3 and rerun the clustering algorithm with a smaller number of clusters. However, if $\tau(k,l) < \infty$ for all pairs of clusters, then we see no evidence of over-clustering and return the inferred cluster assignments to the user.

The recall software package allows users to simulate artificial variables from different classes of null distributions. In the main text, we primarily focus on using a zero-inflated Poisson distribution to generate independent synthetic genes (recall+ZIP). Additional choices that we also consider include generating (1) independent artificial genes via a negative binomial distribution (recall+NB), (2) correlated Poisson-distributed artificial genes via a Gaussian copula (recall+Poisson-copula), (3) correlated NB-distributed artificial genes via a Gaussian copula (recall+NB-copula), and (4) artificial genes drawn using the count splitting method (recall+countsplit).[7] We describe how each of these null distributions is implemented within the recall algorithm and software package in the next section.

Construction of artificial null genes

There has been a large body of work focused on choosing the correct distributions for modeling scRNA-seq count data.[15–18] To construct artificial null genes that "match" the distribution of expression for the original real genes (but without being associated with any particular cell types), we use several approaches, which we detail below.

(C) Bar chart showing the number of clusters detected by the Seurat default and recall when there is a single true group. The Seurat default incorrectly found 4–7 clusters in each simulation, while recall correctly returned a single cluster. The error bars denote the standard deviation.
(D) Demonstration of the traditional clustering framework versus the alternative using recall for simulated data with three known groups. Images left to right show the true labels, clusters found using the Louvain algorithm with default parameter settings in Seurat, and the clusters found using the same Louvain algorithm paired with recall.
(E) Bar chart showing the number of clusters detected by the Seurat default and recall when there are three true groups. The Seurat default incorrectly found 4 clusters in each simulation, while recall correctly returned three clusters. The error bars denote the standard deviation.
(F) Performance comparison of recall, sc-SHC, CHOIR, and scAce on simulated datasets using V-measure. Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of $N = 5{,}000$, 10,000, 25,000, and 50,000 cells. The two recall copula algorithms scaled exponentially with the number of cells and were not able to be completed when $N = 50{,}000$ cells.

Independent ZIP distribution

To construct artificial null genes that match the distribution of expression for the original real genes (but without being associated with any particular cell types), the recall software gives users the choice to implement a univariate parametric modeling approach, which can be applied to each individual gene separately. By default, the algorithm utilizes the ZIP model. Importantly, this parametric generative method creates artificial null gene variables that (1) do not have any association with any particular cell group and (2) do not retain any covariance structure with the original real genes (i.e., the null genes are also independently distributed). The ZIP model mixes two generative processes—the first generates zeros and the second is governed by a Poisson distribution that generates counts (some of which may also be zero).[19] For a random variable $X \sim \text{ZIP}(\pi_0, \lambda)$, we have the following mixture:

$$\begin{aligned} \Pr[X = 0] &= \pi_0 + (1 - \pi_0)\exp\{-\lambda\}, \\ \Pr[X = x] &= (1 - \pi_0)\frac{\lambda^x \exp\{-\lambda\}}{x!}, \end{aligned} \qquad \text{(Equation 3)}$$

where $x \in \mathbb{N}^+$ is any non-negative integer value, λ is the expected count from the Poisson distribution (i.e., the rate parameter), and π_0 is the proportion of extra zeroes arising in addition to those from the underlying Poisson distribution. The maximum likelihood estimators for the ZIP model, given the expression of the j-th gene, take the following form[19] (see Note S1 for full derivation):

$$\widehat{\lambda}_j = W_0\big(-\theta_j \exp\{-\theta_j\}\big) + \theta_j, \widehat{\pi}_{0j} = 1 - \frac{\overline{x}_j}{\widehat{\lambda}_j}, \qquad \text{(Equation 4)}$$

where $\theta_j = \overline{x}_j/(1 - r_{0j})$ represents the sample mean of non-zero counts, $r_{0j} = \sum_i \mathbb{I}(x_{ij} = 0)/N$ denotes the proportion of observed zeroes for the j-th gene across all cells (with $\mathbb{I}(\cdot)$ being an indicator function), $\overline{x}_j$ is the sample average expression for the j-th gene of interest, and W_0 is the principal branch of the Lambert W function (i.e., $W_0(a) = b$ implies $b\exp\{b\} = a$). For each j-th real gene x_j, we fit the maximum likelihood estimators $\widehat{\pi}_{0j}$ and $\widehat{\lambda}_j$ and then sample the synthetic expression for the corresponding artificial null gene as $\tilde{x}_j \sim \text{ZIP}(\widehat{\pi}_{0j}, \widehat{\lambda}_j)$.

Independent NB distribution

As an alternative univariate approach, the recall software also generates independently distributed artificial null genes from a NB model. Briefly, the NB distribution has a probability mass function

$$\Pr[X = k] = \binom{k + r + 1}{k}(1 - p)^k p^r, \qquad \text{(Equation 5)}$$

with the parameter $r \in \mathbb{N}$ representing the number of successes, $p \in [0, 1]$ as the probability of success in each experiment, and $k \in \mathbb{N}_0$ as an integer denoting the number of failures. When both r and p are unknown, there is no analytic solution via maximum likelihood equations. As a result, the recall software estimates the model parameters using the Nelder-Mead algorithm, which is implemented via fitdistr under the MASS R package.[20] When a gene expression vector has a large number of zeroes, the Nelder-Mead algorithm can be unstable. In this scenario, recall will call the fitdist function from the fitdistrplus R package as an attempt to numerically find maximum likelihood estimates for r and p.[21] If this also fails, then the software will estimate r and p via a method of moments. Following this parameter estimation step, the synthetic expression for artificial null genes is sampled as $\tilde{x}_j \sim \text{NB}(\widehat{r}_j, \widehat{p}_j)$.

Correlated Poisson and NB distributions

To generate correlated artificial null genes where each gene marginally follows either a Poisson or NB distribution, recall uses a Gaussian copula as implemented in scDesign3.[22] Copulas are a generalization of inverse transform sampling. Here, the goal is to generate features from any arbitrary distribution by first randomly sampling from a uniform distribution and then transforming those draws via the inverse cumulative distribution function of the distribution of interest. Copulas model the dependence between uniform random variables by applying the probability integral transform to the data. The Gaussian copula is a specific family of distributions over $[0, 1]^G$, where G is the number of artificial null features defined by using the probability integral transform on a multivariate normal distribution of dimension G. For the recall results in the main text, scDesign3 was used to generate correlated count data with either a Poisson or NB marginal distribution where the artificial null genes were assumed to come from a single group.

Countsplit

Countsplit is a method for generating independent train and test splits for a dataset.[7] Again, let N be the number of cells and G the number of genes in a dataset denoted by $\mathbf{X}$. For context, two common approaches in model validation are (1) sample splitting and (2) feature splitting. Sample splitting takes an $N \times J$ matrix and samples two matrices of dimensions $N_1 \times G$ and $N_2 \times G$ (where N_1 is the number of samples in one split and N_2 is the number of samples in the other). Feature splitting, on the other hand, takes an $N \times J$ matrix and creates two matrices that are $N \times G_1$ and $N \times G_2$ (where G_1 is the number of features in one split and G_2 is the number of features in the other). In contrast to these common approaches, countsplit takes an $N \times G$ matrix and samples two matrices that are both of dimension $N \times G$. For data that are assumed to be Poisson distributed, countsplit is performed on a data matrix as follows:

$$x_{ij}^{\text{train}} \sim \text{binomial}(x_{ij}, \epsilon), \qquad \text{(Equation 6)}$$

where $\mathbf{X}^{\text{train}} = [\mathbf{x}_1^{\text{train}}, \ldots, \mathbf{x}_G^{\text{train}}]$, $\mathbf{X}^{\text{test}} = \mathbf{X} - \mathbf{X}^{\text{train}}$, and $0 < \epsilon < 1$. Here, we use the default value of $\epsilon = 0.5$. The key result for countsplit is that if the data are Poisson distributed, then $\mathbf{X}^{\text{test}}$ is independent of $\mathbf{X}^{\text{train}}$. The recall+countsplit algorithm is implemented via the following steps.

(1) Sample $\mathbf{X}^{\text{test}}$ and $\mathbf{X}^{\text{train}}$ from the counts matrix of genes X.
(2) Perform the usual preprocessing on $\mathbf{X}^{\text{train}}$. In this paper, preprocessing consists of normalizing the expression counts followed by PCA.
(3) Apply a given clustering algorithm (e.g., the Louvain algorithm) to the PCA embeddings of the augmented matrix $\mathbf{X}^{\text{train}}$ (or, alternatively, apply the clustering algorithm to the augmented matrix directly).
(4) Conduct differential expression analysis between each k-th and l-th cluster pair, denoted by C_k and C_l, respectively. Obtain p values for all genes (real and artificial nulls) across each comparison.

If, for any pair of clusters, there are no statistically significant genes after the Bonferroni correction, then we return to step 3 and rerun the clustering algorithm with a smaller number of clusters. However, if all pairs of clusters have statistically significant genes in the test set after Bonferroni correction, then the inferred cluster assignments are returned to the user.

Parameters for the recall algorithm

The default starting resolution parameter for the Louvain and Leiden algorithms within recall is $\gamma = 0.8$, the same as the default in the FindClusters function in Seurat. Note that the optimal resolution parameter $\widehat{\gamma}$ is estimated using the combined data frame with both

the real gene expression matrix and the artificial null variables $\mathbf{X}^* = [\mathbf{X}; \tilde{\mathbf{X}}]$. Once this parameter is estimated, recall does not perform any re-clustering on just the original gene expression $\mathbf{X}$ (see rationale in Note S2 and Figure S1). Since recall works by iteratively reducing the starting number of clusters, if the starting resolution parameter is too low (i.e., if you start with correctly calibrated clusters or under-cluster), then there is no opportunity for recall to iteratively reduce the number of clusters. There is a warning produced by the recall software when this occurs, and users can rerun recall with a new parameter to begin with a larger number of clusters.

Simulation studies

For simple proof-of-concept experiments, we simulated scRNA-seq data using the splatter R package,[23] which implements a gamma-Poisson model to create a count matrix for cells. In this study, the one-group dataset was simulated with $G = 1{,}000$ genes and $N = 1{,}000$ cells, while the three-group dataset was simulated to have 1,000 genes and 4,000 cells, with the three groups being separated in proportions of 0.6, 0.2, and 0.2, respectively. Differential gene expression between the groups was controlled using the de.prob parameter with a value of 0.05. For the more comprehensive benchmarking experiments where we compare recall to sc-SHC,[9] CHOIR,[10] and scAce,[11] we additionally simulated five replicates of datasets with 1, 5, and 10 groups each of varying sample sizes in the range $N = 5{,}000$, 10,000, 25,000, and 50,000 cells. The group proportions were drawn from a Dirichlet distribution with concentration parameter $\boldsymbol{\alpha} = (1, 1, \ldots, 1)$. Here, each dataset was simulated with $G = 5{,}000$ genes, and each was given a 10% probability of being differentially expressed between groups. As part of these simulations, we also include an additional study in which we assess the ability to detect rare cell types. Here, we examine two cases where we

(1) vary the number of marker genes while holding the number of rare cells constant and
(2) vary the number of rare cells while holding the proportion of marker genes constant.

In the first scenario, the number of rare cells is fixed at approximately 1% of all simulated cells, while the other "common" cell types are sampled with approximately equal proportions. The rare cells are simulated such that 0.01, 0.02, 0.05, and 0.1 proportion of their genes are differentially expressed; the other cell types were simulated such that a proportion of 0.1 of their genes is differentially expressed. In the second scenario, each cell type is simulated to have the same fixed proportion of differentially expressed genes. The number of rare cells varied as we downsampled the group size to contain 100, 150, and 200 cells. All rare cell type simulations were conducted with $G = 1{,}000$ genes and $N = 5{,}000$ and 10,000 cells.

In order to quantitatively benchmark the performance of recall, countsplit,[7] and ClusterDE[6] on marker gene detection, we repeated simulation 2 above with only two cell types. Using only two cell types simplified this particular analysis and made it straightforward to assess different parameters that would affect method performance. Specifically, we ran the three approaches using the following procedures.

- To run recall: cells were clustered using recall, and then the FindMarkers function in Seurat was used on the resulting cluster pairs (if there were any) to identify differentially expressed genes.
- To run ClusterDE: cells were clustered using the FindClusters function in Seurat, and then ClusterDE was used on the resulting cluster pairs (if there were any) to identify differentially expressed genes.
- To run countsplit: the countsplit training cells were clustered using the FindClusters function in Seurat, and then the FindMarkers function in Seurat was used on the countsplit test cells for each cluster pair (if there were any) to identify differentially expressed genes.

Any genes identified between a pair of clusters were considered "findings" by a particular method—this was to account for the fact that a given clustering algorithm may have over-clustered. In this scenario, under-clustering would result in only 1 cluster, and no marker genes are identified. Each simulated dataset had five replicates and consisted of $N = 1{,}000$ cells and $G = 1{,}000$ genes. We considered three different scenarios where we split the cells into 90/10, 70/30, and 50/50 groups of two. Each cell type was simulated to have 0.1, 0.2, 0.3, 0.4, and 0.5 proportions of its total genes differentially expressed. We simulated each combination of parameters five times. Performance was quantified using precision, recall (sensitivity), and the F1 score.

Data overview

Below, we briefly describe all of the datasets used in this work. All datasets outside of the Tabula Muris were used exclusively to test the scalability of recall and competing methods; therefore, clustering performance was not recorded. All preprocessing steps were done using the Seurat software package. For each of these datasets, the count matrices were log normalized using the NormalizeData function with the default parameters. Here, we set the scale factor = 10,000. The number of variable genes was set to 1,000 for all analyses. This was determined by using the vst selection method implemented by the FindVariableFeatures function. All data were centered and scaled using the ScaleData function with default parameters, principal components were computed with the RunPCA using the variable genes as input, and the nearest-neighbor graphs were computed using the first 10 principal components within the FindNeighbors function. Each evaluated method (recall, sc-SHC, CHOIR, and scAce) was provided with the top 1,000 highly variable genes and the first 10 principal-component embeddings. The implementations of the Louvain clustering algorithms analyzed the nearest-neighbor graphs with resolution values set to $\gamma = 0.8$.

Tabula Muris

To compare the clustering performance of recall against competing methods, we utilized the 20 organs from the Tabula Muris dataset.[24] This dataset contains 53,760 total cells with human-curated cell type labels for each organ. After following the quality control steps outlined in the original study (i.e., filtering to exclude cells with less than 500 total genes detected and to exclude cells with less than 50,000 total reads) and additionally removing cells without a manually curated cell type label, we were left with a total of 45,423 cells for the analysis.

PBMC 3K, Bone Marrow 30K, and Bone Marrow 40K

To assess the runtime and peak memory usage of recall and other competing approaches, we utilized multiple datasets available through the SeuratData R package (web resources). In particular, we downloaded data under the pbmc3k, bmcite, and hcabm40k variable names. For each of these datasets, recall was run with a

larger starting resolution parameter of $\gamma = 1.5$ to ensure that more than one iteration took place.

PBMC 68K

We took scRNA-seq data from fluorescence-activated cell-sorted (FACS) populations of peripheral blood mononuclear cells (PBMCs) provided by Zheng et al.[1] and concatenated each population into one dataset. This dataset contains 68,579 cells with ten different labels corresponding to each purified population that was sorted. The dataset can be found on the 10× Genomics website (web resources).

Liver 8K

This dataset contains 8,444 cells provided by MacParland et al.[25] It can be loaded using the HumanLiver R package (web resources). For this dataset, recall was run with a larger starting resolution parameter of $\gamma = 1.5$ to ensure that more than one iteration took place.

Results

The recall algorithm uses the guiding principle that well-calibrated clusters (i.e., those representing real groups) should have statistically significant differentially expressed genes after correcting for post-selective testing, while over-clustered groups will have much fewer. We use this rule to re-cluster cells iteratively until the inferred clusters are well calibrated and the observed differences in expression between groups are not due to the effects of double dipping. We again emphasize that the main goal of recall is to provide users with the correct number of clusters. The rationale is that estimating the correct number of clusters mitigates the effects of double dipping on post-selective inference for downstream analyses.

As a simple proof of concept, we simulated single-cell gene expression data to compare the clusters found by the widely used Louvain algorithm with default parameter settings in Seurat (with the FindClusters function where the resolution parameter is set to 0.8) versus using the same Louvain algorithm paired with recall. We generated data under two scenarios, each with 1,000 replicates. In the first scenario, there was only one true cell type. Here, the default approach with Seurat incorrectly identified four to seven clusters in all 1,000 replicates (i.e., an error rate of how many times more than a single cluster was identified = 100%), while recall correctly identified only a single cluster (error rate = 0%) (Figures 1B and 1C). In the second scenario, we simulated data such that there were three true cell types. Here, across each of the 1,000 replicates, the Seurat default incorrectly identified four clusters by splitting the larger group into two clusters (i.e., an error rate of how many times more than three clusters were identified = 100%), whereas recall correctly identified three clusters (error rate = 0%) (Figures 1D and 1E).

As a more comprehensive benchmarking study, we next simulated five replicates of datasets with 1, 5, and 10 groups each of varying sample sizes in the range $N = 5{,}000$, 10,000, 25,000, and 50,000 cells. Cell type proportions were drawn from a Dirichlet distribution, which allowed for a mixture between rare and common cell types (see material and methods). We evaluated each strategy for generating artificial null variables within recall and compared them to three recently proposed methods for preventing over-clustering: (1) sc-SHC,[9] (2) CHOIR,[10] and (3) scAce.[11] Both sc-SHC and CHOIR utilize hierarchical clustering paired with permutation tests to decide whether or not to merge clusters, while scAce decides if a pair of clusters should be merged by comparing inter-cluster versus intra-cluster distances. To empirically assess the relative quality of clustering assignments resulting from each method, we utilized common metrics including the adjusted Rand index (ARI), the Jaccard index, the Fowlkes-Mallows index (FMI), the *V*-measure, completeness, and homogeneity.[26] We include a vignette on these cluster evaluation metrics showing their behavior in a simple case study of over-clustering and under-clustering (Note S3; Figure S2). In the main text, we focus our analyses on the ARI due to its popularity in the field[26] and the *V*-measure because it is the harmonic mean between completeness and homogeneity and balances the impact of over-clustering and under-clustering.

The results for each method as evaluated by the *V*-measure (Figure 1F), ARI (Figure S3), completeness (Figure S4), homogeneity (Figure S5), Jaccard index (Figure S6), and FMI (Figure S7) show that recall+ZIP has performance similar to sc-SHC, CHOIR, and scAce in simulation. The recall+ZIP, recall+Poisson-copula, and recall+NB-copula algorithms showed the best performance out of all the recall variants. Performance in terms of computational efficiency and peak memory consumption for each method is shown in Figures S8 and S9, respectively. Here, the recall methods, which simulate artificial genes from independent null distributions, had short runtimes, similar to sc-SHC and scAce. The CHOIR approach was the slowest competing method when datasets were simulated to have 50,000 cells, and the inference for the two recall copula algorithms was so computationally demanding that they failed to even scale to 50,000 cells. Overall, the combined need for both performance and scalability made the independent ZIP the default choice as the artificial variable distribution for recall. Notably, the recall+ZIP model proved to be robust in detecting rare cell types as a function of both the number of marker genes and the total number of rare cells (Figures S10–S15; see material and methods for details). In one scenario where we have sparsely sampled and highly heterogeneous populations (i.e., 5,000 cells with 10 groups), recall+ZIP has a slightly worse ARI and *V*-measure (scoring between 0.85 and 0.90 for both measures) when compared to scAce and CHOIR (scoring between 0.95 and 1.0) but still outperforms sc-SHC.

To evaluate the performance of recall on real scRNA-seq studies, we analyzed 20 different tissues from the Tabula Muris dataset.[24] We again compared recall (implemented with each choice of null distribution for the artificial variables) to sc-SHC, CHOIR, and scAce; all recall results were determined using the Louvain algorithm. We analyzed

the 20 different tissues separately and evaluated the performance of each method by comparing their inferred cluster assignments to the manually curated cell type annotations from the original Tabula Muris study.

When evaluated by the ARI (Figures 2A and S16), *V*-measure (Figures 2B and S17), completeness (Figure S18), homogeneity (Figure S19), Jaccard index (Figure S20), and FMI (Figure S21), recall+ZIP shows state-of-the-art performance. In particular, when evaluated by the ARI, recall+ZIP performs the best in 13 out of the 20 tissues, sc-SHC performs the best in 2 tissues, CHOIR performs the best in 0 tissues, and scAce performs the best in 5 tissues. Similarly, when evaluated by the *V*-measure, recall+ZIP performs the best in 15 tissues, while sc-SHC performs the best in 1 tissue, CHOIR performs the best in 0 tissues, and scAce performs the best in 4 tissues. The clustering results for all algorithms across all 20 tissues are displayed via uniform manifold approximation and projection (UMAP) plots in Figures S22–S41 (for visualization purposes only). For many tissues, CHOIR tended to group cells into many small sub-populations, while for other tissues, sc-SHC severely under-clustered and failed to find any distinct cell types at all, returning only a single group (e.g., aorta, brain myeloid, and pancreas). In the diaphragm tissue, which contains five manually curated cell types, recall+ZIP and sc-SHC matched the five manually curated cell type labels almost exactly, while CHOIR and scAce over-clustered the data with 10 and 8 clusters, respectively (Figure 2C). On the other hand, in the limb muscle dataset, which contains six manually curated cell types, recall+ZIP finds six clusters that closely match the manually curated labels (ARI = 0.97 and *V*-measure = 0.95), while sc-SHC finds 8 clusters (ARI = 0.74 and *V*-measure = 0.79), CHOIR finds 16 clusters (ARI = 0.40 and *V*-measure = 0.69), and scAce finds 11 clusters (ARI = 0.54 and *V*-measure = 0.76) (Figure 2D). In terms of calling clusters exactly according to the curated cell type groups, recall matched the correct number of clusters in 3 tissues (aorta, diaphragm, and limb muscle), and sc-SHC matched the correct number of clusters in 1 tissue (mammary gland), while CHOIR and scAce both matched the correct number of clusters in 0 tissues (Figure S42).

We next investigated the power of each algorithm to detect rare cell types in the Tabula Muris study (Figure S43). We chose to focus on the macrophages in the limb muscle and T cells in the diaphragm because they had the fewest number of cells in their respective tissues ($N = 31$ and 35, respectively) (Figures S43A and S43D). In this experiment, we re-applied each clustering method to each tissue type, with the macrophages and T cells downsampled to have $N = 5$, 10, 15, 25, and 30 cells in the dataset (Figures S43B and S43E). The main goal of this analysis is to assess when each method is no longer able to differentiate the rare cells as being their own individual cluster. When analyzing macrophages in the limb muscle, this threshold was $N = 20$ cells for recall+ZIP and $N = 25$ cells for sc-SHC (Figure S43C). Interestingly, CHOIR and scAce occasionally found more clusters after downsampling the macrophages. In the case of T cells in the diaphragm, while recall+ZIP stopped detecting the rare cell type at $N = 25$ cells, it was still more accurate and stable than sc-SHC, CHOIR, and scAce—all of which consistently, greatly over-clustered in each scenario (Figure S43F). The fact that these methods often over-clustered is a possible explanation for why CHOIR and scAce outperformed recall+ZIP in the rare cell type simulation with 10 cell types, albeit by a small difference in the ARI and *V*-measure (Figures S14 and S15).

Importantly, recall+ZIP exhibited better computational efficiency (i.e., shorter runtime) than the other methods. After running each method on a personal laptop with 6 CPU cores, recall+ZIP was overall the fastest, both sc-SHC and scACe exhibited similarly short runtimes, and CHOIR was the slowest (Figure 2E). For example, in the fat tissue, recall+ZIP finished 0.9 min faster than sc-SHC, 1.4 min faster than scAce, and 15.0 min faster than CHOIR.

To empirically show that the effect of double dipping on downstream tasks is minimized when the correct clusters are estimated, we further compared the clusters determined by the default Seurat implementation of the Louvain algorithm to the clusters determined by the Louvain algorithm with recall+ZIP for the limb muscle tissue in the Tabula Muris study (Figures 3A–3C). Mimicking the typical differential expression workflow, we used the FindMarkers function to identify the top 10 marker genes for each cluster inferred by the Seurat default and recall+ZIP. Qualitatively, the default Louvain implementation appears over-clustered, where inferred clusters 1, 2, 6, and 7 show similar marker gene expression to one another, as do inferred clusters 3 and 5 (Figure 3D). In contrast, the groups found by recall+ZIP show much less shared expression between clusters (Figure 3E). To further investigate whether cells had been over-clustered by the default Louvain algorithm, we performed differential expression analysis between its inferred clusters and observed a high correlation in p values when comparing (1) inferred clusters 1 and 2 versus 3 (Pearson correlation $r = 0.923$) and (2) inferred clusters 1 and 2 versus 5 ($r = 0.925$) (Figure 3F). For the default Louvain algorithm, inferred clusters 1 and 2 both correspond to skeletal muscle satellite cells as annotated by the Tabula Muris Consortium, and inferred clusters 3 and 5 correspond to mesenchymal stem cells. As a comparison, for recall+ZIP, only inferred cluster 1 corresponds to skeletal muscle satellite cells, and only inferred cluster 2 corresponds to mesenchymal stem cells. Differential expression analysis for the recall+ZIP clusters (Figure 3G) results in 506 differentially expressed genes (adjusted p value < 0.05 and an absolute log fold change greater than one), which include many known skeletal muscle satellite cell markers up-regulated in its inferred cluster 1 relative to its inferred cluster 2 (e.g., *Des*, *Chodl*, *Myl12a*, *Asb5*, *Sdc4*, *Apoe*, *Musk*, *Myf5*, *Chrdl2*, and *Notch3*)[27] and mesenchymal stem cell type markers

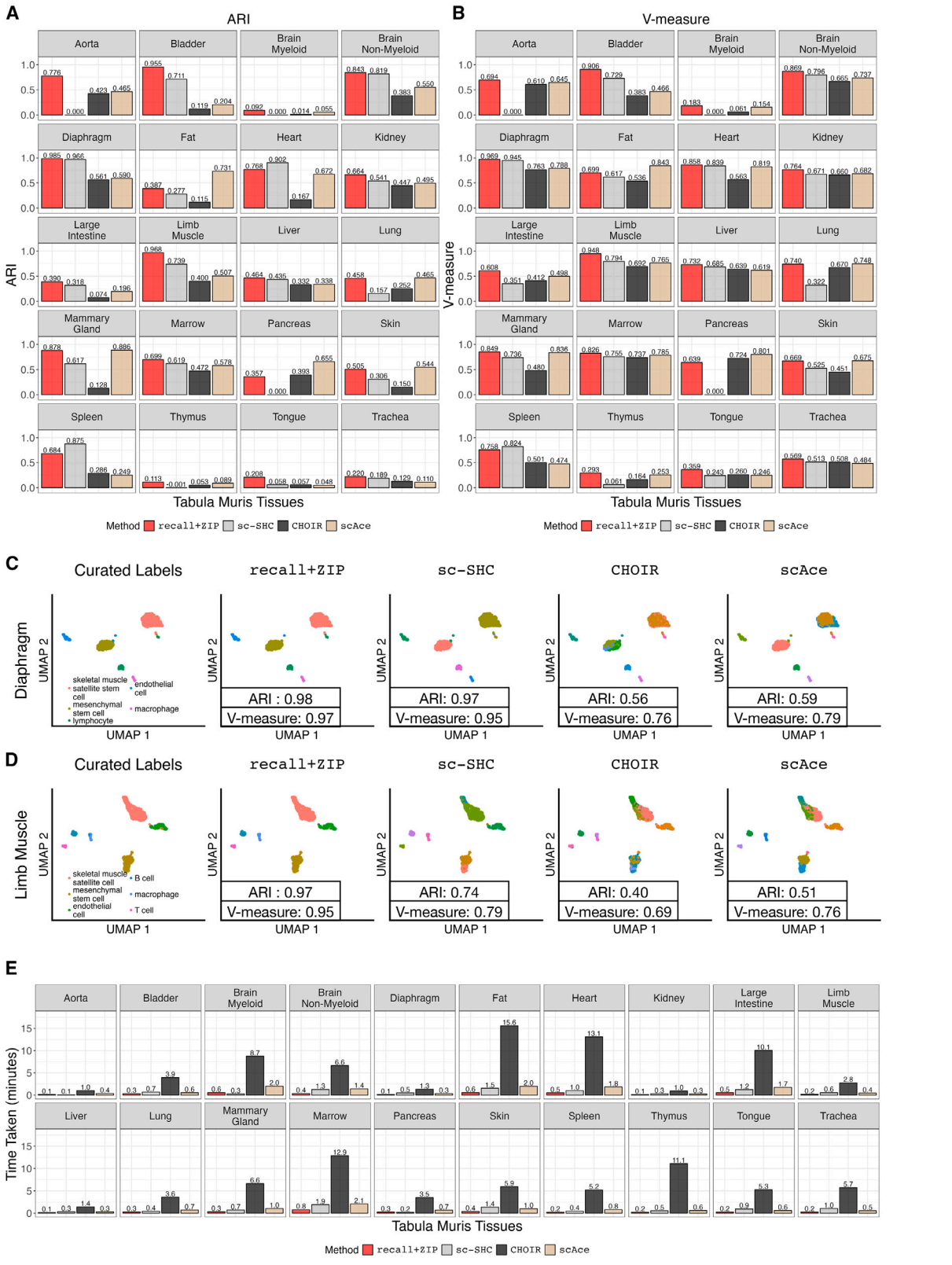


Figure 2. recall+ZIP shows state-of-the-art clustering performance on the Tabula Muris dataset

(A and B) Comparison of recall+ZIP, sc-SHC, CHOIR, and scAce using (A) ARI and (B) *V*-measure for each tissue type in Tabula Muris.

(C and D) Uniform manifold approximation and projection (UMAP) plots displaying the cell type annotations for (C) the diaphragm tissue and (D) the limb muscle tissue datasets. From left to right, we show the manually curated labels from the original study and clusters inferred by recall+ZIP, sc-SHC, CHOIR, and scAce.

(E) Runtime comparison of recall+ZIP, sc-SHC, CHOIR, and scAce for each tissue in the Tabula Muris dataset. Each method was run using 6 CPU cores to emulate the use of a personal laptop.

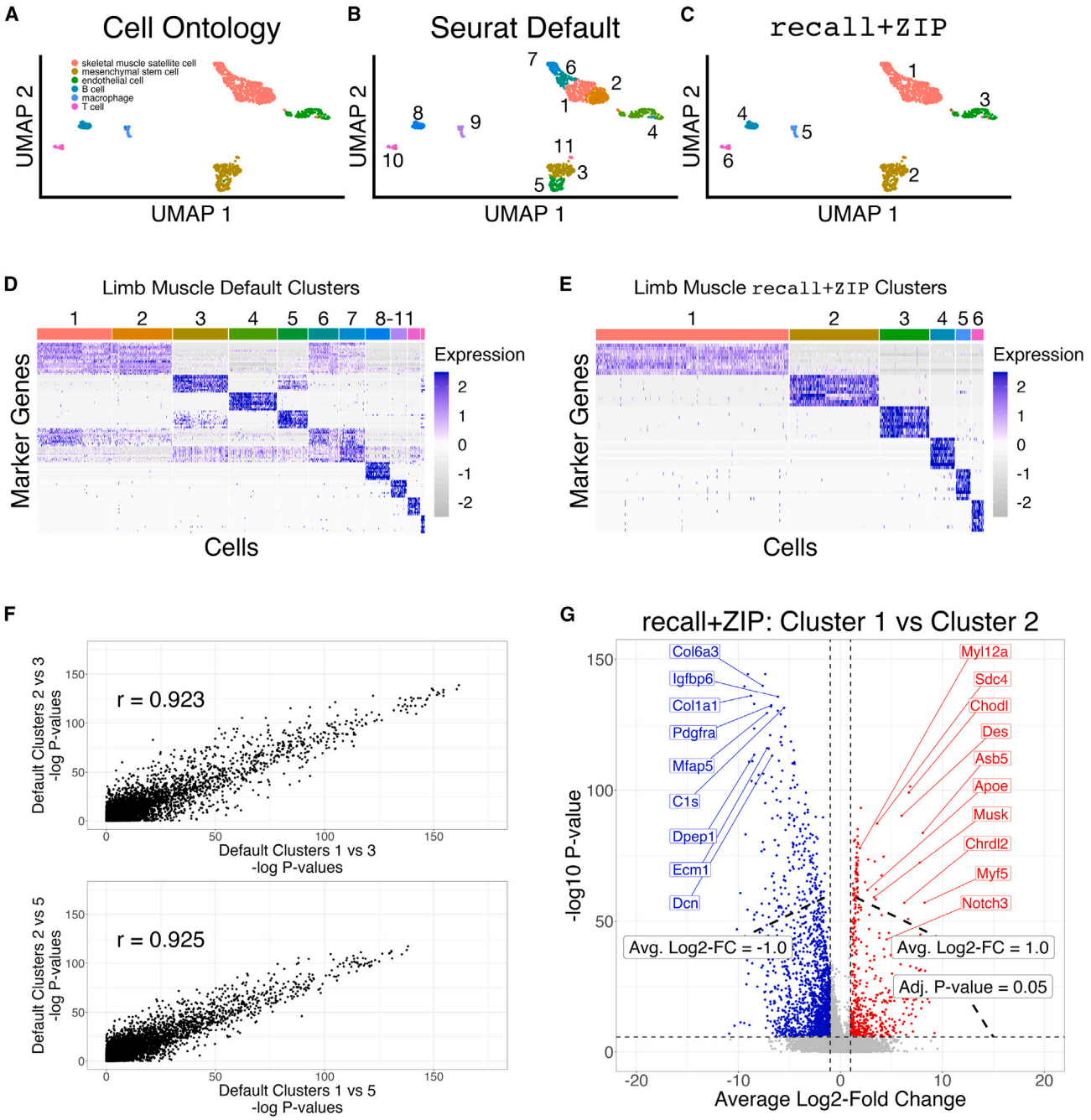


Figure 3. Using recall+ZIP to avoid over-clustering leads to improved hypothesis generation for downstream analyses

(A–C) Uniform manifold approximation and projection (UMAP) plots of the (A) manually curated cell ontology class labels, (B) inferred clusters using the Louvain algorithm with default parameter settings in Seurat, and (C) inferred clusters using the Louvain algorithm paired with recall+ZIP for the limb muscle tissue from the Tabula Muris study.

(D) Heatmap of the top 10 marker genes for each inferred cluster shown in (B) with the default Louvain implementation.

(E) Heatmap of the top 10 marker genes for each inferred cluster shown in (C) with the Louvain algorithm paired with recall+ZIP.

(F) Scatterplots and corresponding Pearson correlation coefficient (r) of the $-\log_{10}$ p values for all genes being tested for differential expression between (1) inferred clusters 1 and 2 versus 3 (top, $r = 0.923$) and (2) inferred clusters 1 and 2 versus 5 (bottom, $r = 0.925$) from (D) using the default Louvain algorithm in Seurat.

(G) Volcano plot of all genes being tested for differential expression between inferred clusters 1 and 2 from (E) using the recall+ZIP version of the Louvain algorithm. The genes colored in red and blue are those with a significant p value after Bonferroni correction and with a $\log_2$ fold change greater than 1 (i.e., up-regulated in cluster 1) or less than -1 (i.e., up-regulated in cluster 2), respectively. The inferred cluster 1 from recall+ZIP corresponds to skeletal muscle satellite cells and cluster 2 corresponds to mesenchymal stem cells. The genes that are labeled are well-known markers of both skeletal muscles (red, up-regulated in cluster 1 relative to cluster 2) and cardiac mesenchymal stem cells (blue, up-regulated in cluster 2 relative to cluster 1).

up-regulated in the inferred cluster 2 relative to the inferred cluster 1 (e.g., *Col6a3*, *Col1a1*, *Igfbp6*, *Pdgfra*, *C1s*, *Mfap5*, *Ecm1*, *Dcn*, and *Dpep1*).[28]

We then benchmarked the quality of marker genes found after calibrating clustering with recall+ZIP versus the genes that were selected after using countsplit[7] and ClusterDE,[6] both of which are differential expression approaches that correct for double dipping. Once again, we assessed the differentially expressed genes between clusters corresponding to skeletal muscle satellite cells and mesenchymal stem cells in the limb muscle tissue (Figure S44A). The countsplit approach generates clusters using a training dataset and then tests for differential expression on a test dataset; alternatively, ClusterDE uses the Seurat default clusters. Both recall and ClusterDE[6] make use of synthetic null variables. The key distinction between these methods is that ClusterDE takes pre-identified cell clusters and computes artificial null data to calibrate statistical null hypothesis tests between those clusters, while recall computes artificial null data on the full dataset first and uses the augmented data matrix as input to the clustering algorithm to calibrate the choice of clusters. The p values obtained by recall+ZIP were highly correlated with those obtained via countsplit ($r = 0.909$; Figure S44B) and moderately correlated with the q values produced by ClusterDE ($r = 0.652$; Figure S44C). Note that the latter result is limited by the fact the most significant genes detected by ClusterDE all have the same q value. Since countsplit has been proven to produce well-calibrated p values,[7] we investigated why the p values produced by recall+ZIP appeared to be inflated (i.e., greater in magnitude on the $-\log$ scale; Figure S44B). Noticing that the recall+ZIP inferred clusters had more cells because the algorithm aims to protect against over-clustering, we randomly downsampled them uniformly to be an analogous size to the clusters identified by countsplit. The p values resulting from these downsampled recall+ZIP clusters were then on the same order of magnitude as the countsplit p values (Figure S44D). This demonstrates that when clusters are indeed well calibrated, marker genes can be detected with increased statistical significance. The high overlap of differentially expressed genes identified by all methods is shown in Figure S44E. There are more marker genes detected by ClusterDE, most likely because it controls false discovery rates at 5%. This is a less stringent criterion than what is applied in recall+ZIP and countsplit, which both use Bonferroni correction to control the family-wise error rate at 5%.

Since this analysis on real data only qualitatively assesses these three methods, we performed an additional set of simulations (described in detail in material and methods) to better evaluate their ability to detect marker genes in a controlled setting (Figure S45). Overall, recall+ZIP and ClusterDE showed very similar performances in terms of precision, recall (i.e., sensitivity), and F1 score across varying cluster sizes and numbers of differentially expressed genes. On the other hand, while countsplit had similar precision, it had a much worse sensitivity and F1 score across the different settings.

As a final analysis of computational scalability, we benchmarked the runtime and peak memory use of recall+ZIP, sc-SHC, CHOIR, and scAce on several other publicly available datasets containing $N = 2{,}700$, 8,444, 30,000, and 40,000 cells (Figures S46 and S47).[25,29,30] Each method was run on a machine with 16 CPU cores, and we ran an additional version of scAce on a GPU. On these datasets, sc-SHC was the fastest, recall+ZIP was a close second, the two scAce implementations were third and fourth, and CHOIR was an order of magnitude slower than all three other methods in last place. Additionally, we applied each method using their default settings on subsets of the 68,579 total PBMCs provided by Zheng et al.[1] as well as on the full dataset. These subsets were of sizes $N = 1{,}000$, 2,000, 5,000, 10,000, 20,000, 30,000, 40,000, 50,000, and 60,000 cells. On these subsets, both recall+ZIP and sc-SHC were very similar in speed, while CHOIR was again an order of magnitude slower (Figure S48). In terms of peak memory consumption, recall+ZIP used the least memory, while sc-SHC showed quadratic memory growth as a function of the number of cells (Figure S49). In summary, recall+ZIP is as fast (or faster) than alternative approaches and uses less memory. Notably, the recall algorithm required less than 10 GB of memory on datasets with nearly 70,000 cells and was able to cluster those cells in less than 15 min with 16 CPU cores. This scalability analysis demonstrates the ability to analyze large datasets with recall on a personal laptop.

Discussion

In conclusion, we present recall, an approach aimed at protecting against over-clustering when analyzing single-cell transcriptomic data. The goal of recall is to provide users with calibrated cluster assignments under the principle that, when correctly inferred, the effect of double dipping on downstream tasks (e.g., differential expression analyses) is minimized. Through the analysis of several large-scale datasets, we demonstrated that recall provides state-of-the-art clustering results at a fraction of the runtime and computer memory when compared to other competing algorithms. Importantly, recall can be efficiently run on a personal laptop when analyzing tens of thousands of cells. We note that cells may exhibit a variety of heterogeneous cell states, continuous axes of variation, or other complexities beyond discrete groups, for which recall, or any clustering algorithm, is not perfectly suited. With its speed and flexibility, recall will save practitioners the hours often spent manually investigating and re-clustering scRNA-seq datasets. We envision that recall will be a useful aid when needing to assign labels to unknown cell types.

The recall approach is not without its limitations. First, the algorithm works downward from an upper bound on the number of clusters (often parameterized by K in the literature). This strategy could potentially lead to

under-clustered results if the starting upper bound is too conservative (i.e., if K is too small). Recall can be initialized with a large set of clusters to circumvent this limitation; however, this will come with an additional computational cost because more iterations will likely need to be performed until the algorithm converges onto a statistically appropriate number of clusters. Second, while the recall software flexibly allows for artificial null genes to be generated from a wide range of probabilistic distributions, the choice of prior distribution is ultimately left up to the user. As shown in both our simulations and analyses on the Tabula Muris dataset, if the synthetic null genes are generated such that the underlying assumptions of the original single-cell data are not met, then recall can be underpowered (Figure 1F). Due to the desire to have a method that is both well powered and scalable, we highlighted the independent ZIP model as the default choice for constructing artificial variables. A key area of our future work is to extend the recall algorithm to select the most appropriate null distribution adaptively in a data-driven way, for example, via model selection by comparing the maximized likelihood of distributions fit over the original data. Third, the current implementation of recall does not account for additional metadata or confounding that might be present in a scRNA-seq dataset. For example, in the presence of batch effects, spurious relationships between cells can be created, and recall might determine that cells of the same type need to be partitioned into different groups (or vice versa). To that end, incorporating data integration steps, like batch effect correction, into the recall software is a relevant direction for future work. One possible extension of the recall algorithm would be to run an integration approach (e.g., Harmony[31]) on the principal-component embeddings of the augmented count matrix to correct for possible confounding before building a K-nearest neighbor graph and performing calibrated clustering.

Data and code availability

All code is available under the open-source MIT license at https://github.com/lcrawlab/recall with documentation at https://lcrawlab.github.io/recall. The scripts used to analyze the data and reproduce the figures of this paper are available at https://github.com/lcrawlab/recallreproducibility. The fully rendered results can also be viewed at https://lcrawlab.github.io/recallreproducibility.

Acknowledgments

We thank members of the Crawford, Shalek, Raghavan, and Winter labs for insightful comments on earlier versions of this manuscript. This research was conducted using computational resources and services provided by the Center for Computation and Visualization at Brown University. This research was also supported in part by a David & Lucile Packard Fellowship for Science and Engineering awarded to L.C. S.R. acknowledges funding support from NCI K08 CA260442. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the funders.

Author contributions

A.D. and L.C. conceived the study and developed the methods. A.D. developed the algorithm and software and led the analyses. A.D., M.L.R., and A.W.N. conducted secondary analyses. S.R., P.S.W., A.P.A., and L.C. co-supervised the project. A.K.S. and L.C. provided resources. A.D. and L.C. wrote the initial draft. All authors interpreted the results and revised the manuscript.

Declaration of interests

S.R. holds equity in Amgen. S.R. and P.S.W. receive research funding from Microsoft. A.P.A. and L.C. are both employees of Microsoft and own equity in Microsoft. P.S.W. reports compensation for consulting from The Engine Ventures unrelated to this work. A.K.S. reports compensation for consulting and/or scientific advisory board membership from Honeycomb Biotechnologies, Cellarity, Ochre Bio, Relation Therapeutics, Fog Pharma, Bio-Rad Laboratories, IntrECate Biotherapeutics, Passkey Therapeutics, and Dahlia Biosciences unrelated to this work.

Supplemental information

Supplemental information can be found online at https://doi.org/10.1016/j.ajhg.2025.02.014.

Web resources

CHOIR, https://github.com/corceslab/CHOIR
ClusterDE, https://github.com/SONGDONGYUAN1994/ClusterDE
countsplit, https://github.com/anna-neufeld/countsplit
Liver 8K dataset, https://github.com/BaderLab/HumanLiver
PBMC 2700 cell dataset, https://cf.10xgenomics.com/samples/cell/pbmc3k/pbmc3k_filtered_gene_bc_matrices.tar.gz
PBMC 68K dataset, https://cf.10xgenomics.com/samples/cell/pbmc68k_rds/pbmc68k_data.rds
recall, https://github.com/lcrawlab/recall
scAce, https://github.com/sldyns/scAce
sc-SHC, https://github.com/igrabski/sc-SHC
SeuratData, https://github.com/satijalab/seurat-data
Tabula Muris dataset, https://figshare.com/articles/dataset/Single-cell_RNA-seq_data_from_Smart-seq2_sequencing_of_FACS_sorted_cells/5715040

Received: February 7, 2025
Accepted: February 12, 2025
Published: March 12, 2025

References

1. Zheng, G.X.Y., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., et al. (2017). Massively parallel digital transcriptional profiling of single cells. Nat. Commun. *8*, 14049. https://doi.org/10.1038/ncomms14049.
2. Macosko, E.Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A.R., Kamitaki, N., Martersteck, E.M., et al. (2015). Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell *161*, 1202–1214. https://doi.org/10.1016/j.cell.2015.05.002.
3. Stoeckius, M., Hafemeister, C., Stephenson, W., Houck-Loomis, B., Chattopadhyay, P.K., Swerdlow, H., Satija, R., and

Smibert, P. (2017). Simultaneous epitope and transcriptome measurement in single cells. Nat. Methods *14*, 865–868. https://doi.org/10.1038/nmeth.4380.
4. Yuhan, H., Stephanie, H., Andersen-Nissen, E., Mauck, W.M., III, Zheng, S., Butler, A., Lee, M.J., Wilk, A.J., Darby, C., Zagar, M., et al. (2021). Integrated analysis of multimodal single-cell data. Cell *184*, 3573–3587. https://doi.org/10.1016/j.cell.2021.04.048.
5. Wolf, F.A., Angerer, P., and Theis, F.J. (2018). Scanpy: large-scale single-cell gene expression data analysis. Genome Biol. *19*, 15. https://doi.org/10.1186/s13059-017-1382-0.
6. Song, D., Li, K., Ge, X., and Li, J.J. (2023). Clusterde: a post-clustering differential expression (de) method robust to false-positive inflation caused by double dipping. Preprint at bioRxiv. https://www.biorxiv.org/content/early/2023/07/25/2023.07.21.550107.
7. Neufeld, A., Gao, L.L., Popp, J., Battle, A., and Witten, D. (2022). Inference after latent variable estimation for single-cell RNA sequencing data. Biostatistics *12*. https://doi.org/10.1093/biostatistics/kxac047.kxac047.
8. Zhang, J.M., Kamath, G.M., and Tse, D.N. (2019). Valid post-clustering differential analysis for single-cell rna-seq. Cell Syst. *9*, 383–392. https://doi.org/10.1016/j.cels.2019.07.012. https://www.sciencedirect.com/science/article/pii/S2405471219302698.
9. Grabski, I.N., Kelly, S., and Irizarry, R.A. (2023). Significance analysis for clustering with single-cell rna-sequencing data. Nat. Methods *20*, 1548–7105. https://doi.org/10.1038/s41592-023-01933-9.
10. Petersen, C., Mucke, L., and Corces, M.R. (2024). Choir improves significance-based detection of cell types and states from single-cell data. Preprint at bioRxiv. https://doi.org/10.1101/2024.01.18.576317.
11. He, X., Qian, K., Wang, Z., Zeng, S., Li, H., and Li, W.V. (2023). scAce: an adaptive embedding and clustering method for single-cell gene expression data. Bioinformatics *39*, btad546. https://doi.org/10.1093/bioinformatics/btad546.
12. Foygel Barber, R., and Candès, E.J. (2015). Controlling the false discovery rate via knockoffs. Ann. Stat. *43*, 2055–2085. https://doi.org/10.1214/15-aos1337.
13. Candès, E., Fan, Y., Janson, L., and Lv, J. (2018). Panning for gold: 'model-x' knockoffs for high dimensional controlled variable selection. J. Roy. Stat. Soc. B *80*, 551–577. https://doi.org/10.1111/rssb.12265.
14. Korsunsky, I., Nathan, A., Millard, N., and Raychaudhuri, S. (2019). Presto scales wilcoxon and auroc analyses to millions of observations. Preprint at bioRxiv. https://doi.org/10.1101/653253.
15. Sarkar, A., and Stephens, M. (2021). Separating measurement and expression models clarifies confusion in single-cell rna sequencing analysis. Nat. Genet. *53*, 770–777. https://doi.org/10.1038/s41588-021-00873-4.
16. Svensson, V. (2020). Droplet scrna-seq is not zero-inflated. Nat. Biotechnol. *38*, 147–150. https://doi.org/10.1038/s41587-019-0379-5.
17. Kharchenko, P.V., Silberstein, L., and Scadden, D.T. (2014). Bayesian approach to single-cell differential expression analysis. Nat. Methods *11*, 740–742. https://doi.org/10.1038/nmeth.2967.
18. Ahlmann-Eltze, C., and Huber, W. (2021). glmGamPoi: fitting Gamma-Poisson generalized linear models on single cell count data. Bioinformatics *36*, 5701–5702. https://doi.org/10.1093/bioinformatics/btaa1009.
19. Dencks, S., Piepenbrock, M., and Schmitz, G. (2020). Assessing vessel reconstruction in ultrasound localization microscopy by maximum likelihood estimation of a zero-inflated poisson model. IEEE Trans. Ultrason. Ferroelectrics Freq. Control *67*, 1603–1612. https://doi.org/10.1109/TUFFC.2020.2980063.
20. Venables, W.N., and Ripley, B.D. (2002). Modern Applied Statistics with S, fourth edition (Springer). https://www.stats.ox.ac.uk/pub/MASS4/.
21. Delignette-Muller, M.L., and Dutang, C. (2015). fitdistrplus: An R package for fitting distributions. J. Stat. Software *64*, 1–34. https://doi.org/10.18637/jss.v064.i04.
22. Song, D., Wang, Q., Yan, G., Liu, T., Sun, T., and Li, J.J. (2024). scdesign3 generates realistic in silico data for multimodal single-cell and spatial omics. Nat. Biotechnol. *42*, 247–252. https://doi.org/10.1038/s41587-023-01772-1.
23. Zappia, L., Phipson, B., and Oshlack, A. (2017). Splatter: simulation of single-cell rna sequencing data. Genome Biol. *18*, 174. https://doi.org/10.1186/s13059-017-1305-0.
24. Tabula Muris Consortium; Overall coordination; Logistical coordination; Organ collection and processing; Library preparation and sequencing; Computational data analysis; Cell type annotation; Writing group; Supplemental text writing group; and Principal investigators (2018). Single-cell transcriptomics of 20 mouse organs creates a tabula muris. Nature *562*, 367–372. https://doi.org/10.1038/s41586-018-0590-4.
25. MacParland, S.A., Liu, J.C., Ma, X.-Z., Innes, B.T., Bartczak, A.M., Gage, B.K., Manuel, J., Khuu, N., Echeverri, J., Linares, I., et al. (2018). Single cell rna sequencing of human liver reveals distinct intrahepatic macrophage populations. Nat. Commun. *9*, 4383. https://doi.org/10.1038/s41467-018-06318-7.
26. Yu, L., Cao, Y., Yang, J.Y.H., and Yang, P. (2022). Benchmarking clustering algorithms on estimating the number of cell types from single-cell rna-sequencing data. Genome Biol. *23*, 49. https://doi.org/10.1186/s13059-022-02622-0.
27. De Micheli, A.J., Laurilliard, E.J., Heinke, C.L., Ravichandran, H., Fraczek, P., Soueid-Baumgarten, S., De Vlaminck, I., Elemento, O., and Cosgrove, B.D. (2020). Single-Cell analysis of the muscle stem cell hierarchy identifies heterotypic communication signals involved in skeletal muscle regeneration. Cell Rep. *30*, 3583–3595.
28. Pisterzi, P., Chen, L., van Dijk, C., Wevers, M.J.W., Bindels, E.J.M., and Raaijmakers, M.H.G.P. (2023). Resource: A cellular developmental taxonomy of the bone marrow mesenchymal stem cell population in mice. Hemasphere *7*, e823.
29. Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W.M., Hao, Y., Stoeckius, M., Smibert, P., and Satija, R. (2019). Comprehensive integration of single-cell data. Cell *177*, 1888–1902. https://doi.org/10.1016/j.cell.2019.05.031.
30. Regev, A., Teichmann, S.A., Lander, E.S., Amit, I., Benoist, C., Birney, E., Bodenmiller, B., Campbell, P., Carninci, P., Clatworthy, M., Clevers, H., et al. (2017). Science forum: The human cell atlas. Elife *6*, e27041. https://doi.org/10.7554/eLife.27041.
31. Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.-ru, and Raychaudhuri, S. (2019). Fast, sensitive and accurate integration of single-cell data with harmony. Nat. Methods *16*, 1289–1296. https://doi.org/10.1038/s41592-019-0619-0.

The American Journal of Human Genetics, Volume *112*

Supplemental information

Artificial variables help to avoid over-clustering in single-cell RNA sequencing

Alan DenAdel, Michelle L. Ramseier, Andrew W. Navia, Alex K. Shalek, Srivatsan Raghavan, Peter S. Winter, Ava P. Amini, and Lorin Crawford

Supplemental Note 1

In this Supplemental Note, we derive the maximum likelihood estimates (MLE) for the zero-inflated Poisson (ZIP). The ZIP model mixes two generative processes—the first generates zeros and the second is governed by a Poisson distribution that generates counts (some of which may also be zero). For a random variable $X \sim \text{ZIP}(\pi_0, \lambda)$, the probability mass function is

$$\Pr[X = 0] = \pi_0 + (1 - \pi_0)\exp\{-\lambda\}, \qquad \Pr[X = x] = (1 - \pi_0)\frac{\lambda^x \exp\{-\lambda\}}{x!} \tag{1}$$

where $x \in \mathbb{N}^+$ is any non-negative integer value, λ is the expected count from the Poisson distribution (i.e., the rate parameter), and π_0 is the proportion of extra zeroes arising in addition to those from the underlying Poisson distribution. The log-likelihood equation for the ZIP model, given the expression of the j-th gene, takes the following form

$$\ell(\pi_0, \lambda; x_1, \ldots, x_n) = \sum_{i:x_i=0} \log\left[\pi_0 + (1 - \pi_0)\exp\{-\lambda\}\right] + \sum_{i:x_i\neq 0} \log\left[(1 - \pi_0)\frac{\lambda^x \exp\{-\lambda\}}{x!}\right]. \tag{2}$$

We now derive the MLEs for unknown parameters π_0 and λ. Let $r_0 = \sum_i \mathbb{I}(x_i = 0)/n$ denote the proportion of observed zeroes for a given feature across all samples (with $\mathbb{I}(\bullet)$ being an indicator function). Notice that Eq. (2) simplifies to the following

$$\ell(\pi_0, \lambda; x_1, \ldots, x_n) = nr_0 \log\left[\pi_0 + (1 - \pi_0)\exp\{-\lambda\}\right] + n(1 - r_0)[\log(1 - \pi_0) - \lambda] + n\bar{x}\log\lambda. \tag{3}$$

We now can find the closed form estimates for both parameters in two parts.

Estimating equation for λ

To begin, we take the partial derivative of Eq. (3) with respect to π_0. This takes the following form

$$\frac{\partial \ell}{\partial \pi_0} = \frac{nr_0(1 - \exp\{-\lambda\})}{\pi_0 + (1 - \pi_0)\exp\{-\lambda\}} - \frac{n(1 - r_0)}{1 - \pi_0}. \tag{4}$$

Next, we set this expression equal to 0 and simplify such that

$$nr_0(1 - \exp\{-\lambda\})(1 - \pi_0) - n(1 - r_0)(\pi_0 + (1 - \pi_0)\exp\{-\lambda\}) = 0. \tag{5}$$

We then may solve for π_0 and get the follow expression

$$\pi_0 = \frac{\exp\{\lambda\}r_0 - 1}{\exp\{\lambda\} - 1}. \tag{6}$$

Treating π_0 as a nuisance parameter, we can substitute this back into the log-likelihood in Eq. (3) to obtain the profile log-likelihood for λ where

$$\ell_p(\ell(\pi_0, \lambda; x_1, \ldots, x_n, \pi_0) = -n(1 - r_0)[\lambda + \log(1 - \exp\{-\lambda\})] + n\bar{x}\log\lambda + \text{constant}. \tag{7}$$

Taking the derivative of the profile log-likelihood in Eq. (7), with respect to λ, yields

$$\frac{\partial \ell_p}{\partial \lambda} = \frac{\exp\{\lambda\}n(1 - r_0)}{\exp\{\lambda\} - 1} + \frac{n\bar{x}}{\lambda}. \tag{8}$$

Next, we set this expression equal to 0 and simplify such that

$$0 = (1 - r_0)\lambda + \overline{x}(1 - \exp\{-\lambda\}) \quad \Longleftrightarrow \quad \bar{x}(1 - \exp\{-\lambda\}) = \lambda(1 - r_0) \tag{9}$$

which is widely reported in the literature [6, 7]. Typically, it is claimed that the right hand expression of Eq. (9) must be solved numerically; however, it was recently shown that this equation can be solved using the principal branch of the Lambert W function [7]. Let $\theta = \bar{x}/(1 - r_0)$ and W_0 be the principal branch of the Lambert W function (i.e., $W_0(a) = b$ implies $b \exp\{b\} = a$). In this case, Eq. (9) is then

$$\theta = \frac{\lambda}{1 - \exp\{-\lambda\}}. \tag{10}$$

The key now is to isolate λ. We start making the following transformation

$$\lambda = \theta(1 - \exp\{-\lambda\}). \tag{11}$$

Next, we subtract θ from both sides of the equation and simplify the expression to be

$$\lambda - \theta = -\theta \exp\{-\lambda\}. \tag{12}$$

Then, we multiply both sides of the expression by $\exp\{\lambda - \theta\}$ such that

$$(\lambda - \theta) \exp\{\lambda - \theta\} = -\theta \exp\{-\theta\}. \tag{13}$$

It is important to notice that this the above is of the form $b \exp\{b\} = a$ and can be inverted using the principal branch of the Lambert W function to obtain

$$\lambda - \theta = W_0(-\theta \exp\{-\theta\}). \tag{14}$$

Finally, this gives us the MLE as $\hat{\lambda} = W_0(-\theta \exp\{-\theta\}) + \theta$.

Estimating equation for π_0

Remember in Eq. (6), we showed that the partial derivative of the log-likelihood in Eq. (3) with respect to π_0 simplified to

$$\pi_0 = \frac{\exp\{\lambda\} r_0 - 1}{\exp\{\lambda\} - 1}. \tag{15}$$

Furthermore, using the profile likelihood in Eq. (9), we also derived that

$$\bar{x}(1 - \exp\{-\lambda\}) = \lambda(1 - r_0) \quad \Longleftrightarrow \quad \frac{\bar{x}}{\lambda} = \frac{1 - r_0}{1 - \exp\{-\lambda\}}. \tag{16}$$

Subtracting by 1 on both sides of Eq. (16) and simplifying, we can rearrange this expression to be

$$\begin{aligned} 1 - \frac{\overline{x}}{\lambda} &= 1 - \frac{1 - r_0}{1 - \exp\{-\lambda\}} \\ &= 1 - \frac{\exp\{\lambda\} - \exp\{\lambda\} r_0}{\exp\{\lambda\} - 1} \\ &= \frac{\exp\{\lambda\} r_0 - 1}{\exp\{\lambda\} - 1} \\ &= \pi_0 \end{aligned} \tag{17}$$

Therefore, the maximum likelihood estimate is $\hat{\pi}_0 = 1 - \bar{x}/\hat{\lambda}$.

Supplemental Note 2

Consider a study with single-cell RNA sequencing (scRNA-seq) expression data for $i = 1, \ldots, N$ cells that each have measurements for $j = 1, \ldots, G$ genes. Let this dataset be represented by an $N \times G$ matrix $\mathbf{X}$ where the column-vector $\mathbf{x}_j$ denotes the expression profile for the j-th gene. In the `recall` framework, we generate an artificial null expression vector $\tilde{\mathbf{x}}_j$ for each gene in the study $\mathbf{x}_j$. Next, we concatenate all of the synthetic genes together and construct a matrix of artificial null variables $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_G]$. During an analysis, `recall` finds the optimal resolution parameter for the Louvain algorithm (denoted as γ in the main text) by using the combined data frame containing both the real gene expression matrix and the artificial null variables $\mathbf{X}^* = [\mathbf{X}; \tilde{\mathbf{X}}]$. In this Supplemental Note, we demonstrate why `recall` does not perform any re-clustering on just the original gene expression $\mathbf{X}$ once the resolution parameter is estimated. To do this, we use a small scRNA-seq study of peripheral blood mononuclear cells (PBMCs) from 10x Genomics (via reproducible `R` code below). We begin by loading in the required packages and the single-cell dataset.

```
### Set seed for reproducibility ###
set.seed(1234)

### Load in libraries and packages ###
library(Seurat)
library(recall)
library(SeuratData)
library(ggplot2)
library(recallreproducibility)

### Load in data ###
data("pbmc3k")
```

Next, we process the single-cell dataset with a typical bioinformatic workflow using `Seurat` [8].

```
pbmc3k <- UpdateSeuratObject(pbmc3k)

pbmc3k <- NormalizeData(pbmc3k)
pbmc3k <- FindVariableFeatures(pbmc3k)
pbmc3k <- ScaleData(pbmc3k)
pbmc3k <- RunPCA(pbmc3k)
pbmc3k <- FindNeighbors(pbmc3k)
pbmc3k <- RunUMAP(pbmc3k, dims = 1:10)
```

Now, we run a calibrated clustering analysis using `recall` on the concatenated data $\tilde{\mathbf{X}}$.

```
pbmc_recall <- FindClustersRecall(pbmc3k)
```

The optimal resolution parameter used by `recall` for the Louvain algorithm was $\gamma = 0.4096$. To demonstrate the main point of this vignette, we feed this resolution parameter γ back in the Louvain algorithm and allow it to perform clustering on the original gene expression data $\mathbf{X}$ only.

```
pbmc_default <- FindClusters(pbmc3k, resolution = 0.4096)
```

Finally, we plot uniform manifold approximation and projection (UMAP) plots with each of the cluster labels applied (for visualization purposes only).

```
umap_recall <- custom_scatter(pbmc_recall,
                              "umap",
                              group_by = "recall_clusters",
                              x_title = "UMAP 1",
                              y_title = "UMAP 2",
                              pt.size = 2) +
                              Seurat::NoLegend()

umap_default <- custom_scatter(pbmc_default,
                               "umap",
                               group_by = "seurat_clusters",
                               x_title = "UMAP 1",
                               y_title = "UMAP 2",
                               pt.size = 2) +
                               Seurat::NoLegend()

cl1 <- patchwork::wrap_elements(panel =
                                  grid::textGrob('Initial clustering\nwith recall',
                                                 gp = grid::gpar(fontsize = 64)))

cl2 <- patchwork::wrap_elements(panel =
                                  grid::textGrob('Re-clustering with
                                                 \nnew recall resolution',
                                                 gp = grid::gpar(fontsize = 64)))
comparison_umap <- cl1 + cl2 +
  umap_recall +  umap_default +
  patchwork::plot_layout(widths = c(5, 5),
                         heights = c(1,3))

ggsave("rerun_clustering.png", comparison_umap, dpi = 600,
       width = 24, height = 12, units = "in")
```

As evident in Fig. S1, the clusters obtained by each method (even with the same resolution parameter) are different — this is because the optimization ultimately happens on different sets of input data. Specifically, there were 8 clusters identified by `recall` on the combined data frame containing both the real gene expression matrix and the artificial null variables $\mathbf{X}^* = [\mathbf{X}; \tilde{\mathbf{X}}]$, but 9 clusters identified by the Louvain algorithm with the same resolution when run only on the original data $\mathbf{X}$.

Supplemental Note 3

In this Supplemental Note, we detail the metrics used to evaluate the quality of clustering assignments against ground truth, which are referred to as extrinsic clustering metrics. For the analyses conducted in the main text, we use the manually curated cell type annotations from the original studies as a proxy for ground truth labels. All of the metrics described here (except for the adjusted Rand index) vary between 0 (poor agreement between inferred cluster assignments and ground truth) and 1 (good agreement between inferred cluster assignments and ground truth). The adjusted Rand index (ARI) ranges between [-1/2, 1] where 1 represents perfect agreement between label sets, 0 represents random agreement, and negative values represent worse than random agreement [1]. There are two broad classes of clustering evaluation metrics utilizing ground truth labels: (1) confusion matrix-based metrics which are based on notions of true positives, true negatives, false positives, and false negatives, and (2) entropy-based metrics which measure the uncertainty in inferred clustering assignments given a set of reference labels (and vice versa).

Confusion Matrix-Based Cluster Evaluation Metrics

The typical confusion matrix is defined as follows for a classification problem with 2 classes.

		True Label		
		Class 1	Class 2	Total
Inferred Label	Class 1	TP	FP	a_1
	Class 2	FN	TN	a_2
	Total	b_1	b_2	n

where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, and FN is the number of false negatives. This can be similarly defined for a classification problem with an arbitrary number of classes. Since a clustering result may have a different number of inferred clusters than the number of true labels and since it is not immediately obvious how to know which true label an inferred cluster might correspond to, we instead define these metrics over pairs of points in the following way:

- True positives (TP) is the number of cell pairs that have the **same** true labels and are assigned the **same** inferred labels after clustering.
- True negatives (TN) is the number of cell pairs that have **different** true labels and also have **different** inferred labels after clustering.
- False positives (FP) is the number cell pairs that have **different** true labels but are assigned the **same** inferred labels after clustering.
- False negatives (FN) is the number of cell pairs that have the **same** true labels but are assigned **different** inferred labels after clustering.

We describe a series of confusion matrix-based cluster evaluation metrics below using these concepts. To simplify notation, we will let n_{ij}, a_i, and b_j be values obtained from a contingency table, and $n = \sum_{ij} n_{ij}$ for indices i and $j \in \{1, 2\}$. To be specific, TP $= n_{11}$, FP $= n_{12}$, FN $= n_{21}$, and TN $= n_{22}$, respectively. An illustration of these metrics can be found in Fig. S2.

Adjusted Rand Index (ARI)

The adjusted Rand index (ARI) captures the similarity between labels inferred by a clustering algorithm and the reference labels. It is based on the Rand index (RI) which is computed as the following

$$\text{RI} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} = \frac{b_1}{n}. \tag{18}$$

The ARI corrects for the RI measurement's sensitivity to chance via permutation [2] where

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{1 - \mathbb{E}[\text{RI}]} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \binom{n}{2}}{1/2 \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \binom{n}{2}}. \tag{19}$$

Here, $\mathbb{E}[\text{RI}]$ denotes the expectation of the Rand index.

Jaccard Similarity Index

For high-dimensional single-cell studies with a large number of both cells and cell types, the number of true negatives can be quite large. The Jaccard similarity index captures the amount of overlap between two finite sets. Overall, the Jaccard index can be a useful alternative to the ARI because it ignores the contribution of the true negatives. Formally, it is defined as the following

$$J = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} = \frac{n_{11}}{a_1 + n_{21}} \tag{20}$$

which ranges between [0, 1] where 1 denotes complete overlap between the inferred and true cluster labels, and 0 signifies no overlap.

Folkes-Mallows Index (FMI)

The Folkes-Mallows Index is the geometric mean of the positive predictive value (PPV) and the true positive rate (TPR) [3]. More concretely, these values are computed via the following

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{n_{11}}{a_1} \qquad \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{n_{11}}{b_1} \tag{21}$$

A higher value for the Fowlkes–Mallows index indicates a greater similarity between the inferred clusters and the true labels. This is computed by

$$\text{FMI} = \sqrt{\text{PPV} \times \text{PPR}}. \tag{22}$$

The FMI is also on the unit scale ranging from [0, 1] where 0 corresponds to the worst inferred clusters such that both inferred and true groups are completely unrelated, and 1 corresponds to the scenario in which both groupings perfectly agree.

Entropy-Based Cluster Evaluation Metrics

Entropy-based clustering metrics aim to quantify the amount of information shared between the inferred label distribution and the reference label distribution. We describe a series of these metrics below.

Completeness

Completeness describes the uncertainty in the inferred clustering assignment conditioned on the true labels where 1 is a "good" value for satisfying completeness. In other words, it quantifies how often cells of the same type are also in the same cluster. Formally,

$$C = 1 - \frac{\mathbb{H}(\mathcal{A} \,|\, \mathcal{B})}{\mathbb{H}(\mathcal{A})} \tag{23}$$

where $\mathcal{B}$ represents the set of true labels, $\mathcal{A}$ represents the inferred clustering assignments, and $\mathbb{H}(\cdot)$ represents the Shannon entropy of a given label distribution [4]. When all cells from true group B belong to the same inferred cluster, the completeness metric equals 1. For the degenerate case where $\mathbb{H}(\mathcal{A}, \mathcal{B}) = 0$, we define $C = 0$.

Homogeneity

Homogeneity describes the uncertainty in the true label conditioned on the cluster assignment where 1 is a "good" value for satisfying homogeneity. In other words, it quantifies the amount that cells within a cluster are also from the same cell type. Formally, homogeneity is defined

$$H = 1 - \frac{\mathbb{H}(\mathcal{B} \,|\, \mathcal{A})}{\mathbb{H}(\mathcal{B})} \tag{24}$$

where, again, $\mathcal{B}$ represents the set of true labels, $\mathcal{A}$ represents the inferred clustering assignments, and $\mathbb{H}(\cdot)$ represents the Shannon entropy of a given label distribution [4]. When all cells from all of the clusters contain only cells that are from the same true group, the homogeneity metric equals 1. For the degenerate case where $\mathbb{H}(\mathcal{A}, \mathcal{B}) = 0$, we define $H = 0$.

V-Measure Balances Between Completeness and Homogeneity

A key observation when using completeness and homogeneity to evaluate clustering algorithms is that they tend to penalize over- and under-clustering, respectively.

- Consider a simple example of over-clustering where there is a group of cells that is split in half into two clusters. Then the uncertainty in the inferred clustering assignments $\mathcal{A}$, given the true labels $\mathcal{B}$, is high because half of the group is in each cluster. Formally, this means that the Shannon entropy $\mathbb{H}(\mathcal{A} \,|\, \mathcal{B})$ is high and so the completeness will be low.
- On the other hand, consider a simple example of under-clustering where two different cell groups are assigned to the same cluster. Then the uncertainty in the true group labels $\mathcal{B}$, given the inferred clustering assignments $\mathcal{A}$, is high because the cells in the cluster could come from either of the two groups. Formally, the entropy $\mathbb{H}(\mathcal{B} \,|\, \mathcal{A})$ is high and so homogeneity will be low.

The V-measure is defined as the weighted harmonic mean of completeness and homogeneity where

$$V(\beta) = \frac{(1+\beta)HC}{\beta H + C} \tag{25}$$

where β dictates the contribution of completeness versus homogeneity [4]. In this paper, we set $\beta = 1$ such that the two are weighted equally and the V-measure is simply

$$V(1) = \frac{2HC}{H + C} \tag{26}$$

which is the simple harmonic mean of completeness and homogeneity. The V-measure ranges between [0, 1] where 1 represents total information sharing between label sets and 0 represents no information sharing between label sets. It requires that both homogeneity and completeness are maximized and will be zero if a clustering algorithm completely fails to satisfy either of the two properties.

Simulation Study: Behavior of Cluster Evaluation Metrics

In order to quantitatively demonstrate the behavior of both the confusion matrix-based and entropy-based clustering metrics, we consider a synthetic dataset with three groups of cells (see Fig. S2). When an algorithm under-clusters, it will merge two of the clusters; while, when an algorithm over-clusters, it will infer a fourth group made up of cells from two of the true groups. The table below shows the behavior of different clustering evaluation metrics in the cases when an algorithm finds the true groups, under-clusters, and over-clusters.

Metric	True Groups	Under-Clustered	Over-Clustered
ARI	1.0	0.55	0.83
Jaccard index	1.0	0.58	0.78
FMI	1.0	0.76	0.88
Homogeneity	1.0	0.54	0.94
Completeness	1.0	0.92	0.77
V-measure	1.0	0.68	0.85

Notice that ARI, Jaccard index, and homogeneity qualitatively perform similarly by penalizing under-clustering more than over-clustering. In contrast, completeness penalizes over-clustering and rewards under-clustering. FMI and V-measure balance between both metrics. For this reason, in the main text, we focus on ARI due to its popularity in the literature [5] and V-measure because of its ability to balance the impact of over-clustering and under-clustering when evaluating `recall`, `sc-SHC`, `CHOIR`, and `scAce`.

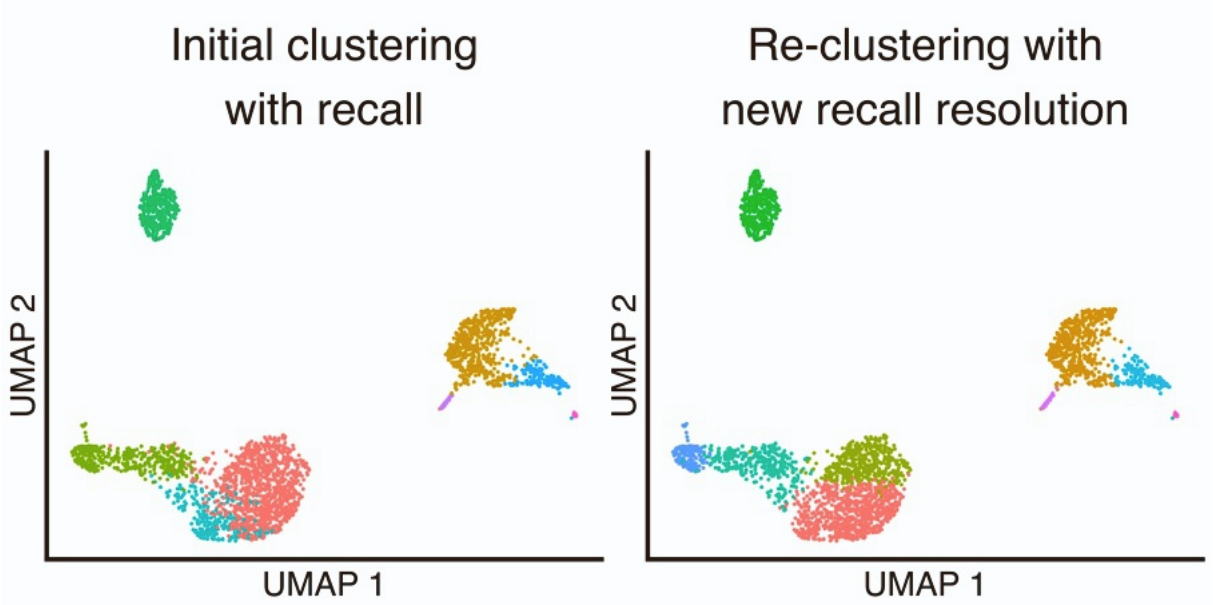


Figure S1. Uniform manifold approximation and projection (UMAP) plots comparing clustering results when the original gene expression data is reclustered using the resolution parameter estimated by `recall`. On the left hand side, clustering is shown for the normal workflow where `recall` finds the optimal resolution parameter for the Louvain algorithm (denoted as γ in the main text) by using the combined data frame containing both the real gene expression matrix and the artificial null variables $\mathbf{X}^* = [\mathbf{X}; \tilde{\mathbf{X}}]$. On the right hand side, the Louvain algorithm re-clusters on just the original gene expression $\mathbf{X}$ once the resolution parameter is estimated via `recall`. The clusters obtained by each method (even with the same resolution parameter) are different — this is because the optimization ultimately happens on different sets of input data. Specifically, there were 8 clusters identified by `recall` on the combined data frame (used for controlling calibration), but 9 clusters identified by the Louvain algorithm with the same resolution when run only on the original data.

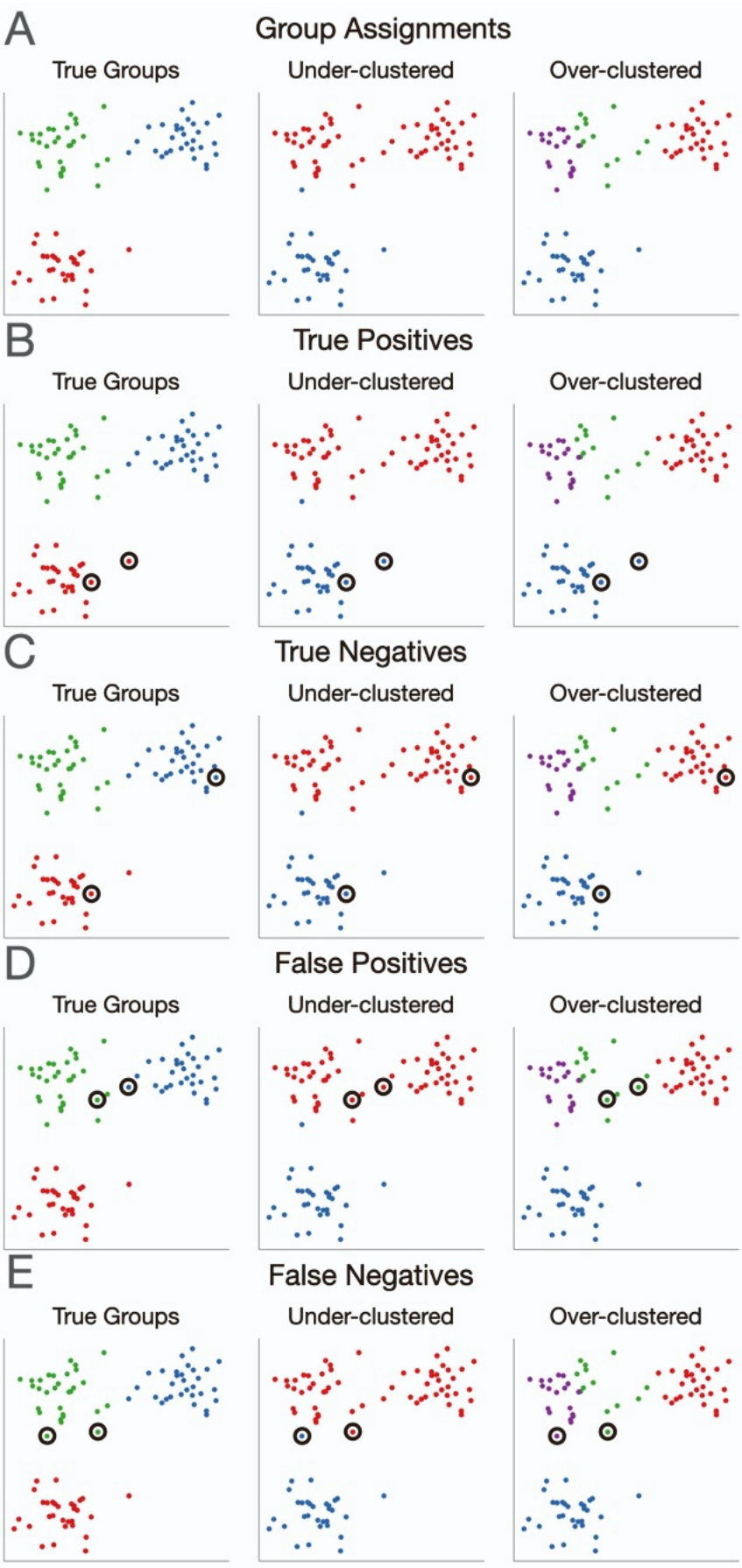


Figure S2. (Continued on the following page).

Figure S2. Demonstration of cases of over- and under-clustering in single-cell analyses. Here, we generate synthetic data from a Gaussian mixture model. Data are created such that there are three true groups. Panel **(A)** shows a depiction of the true cluster labels and the inferred cluster assignments when an algorithm under- and over-clusters, respectively. For comparison, we also show an example of definitions for **(B)** true positives, **(C)** true negatives, **(D)** false positives, and **(E)** false negatives used by confusion matrix-based metrics.

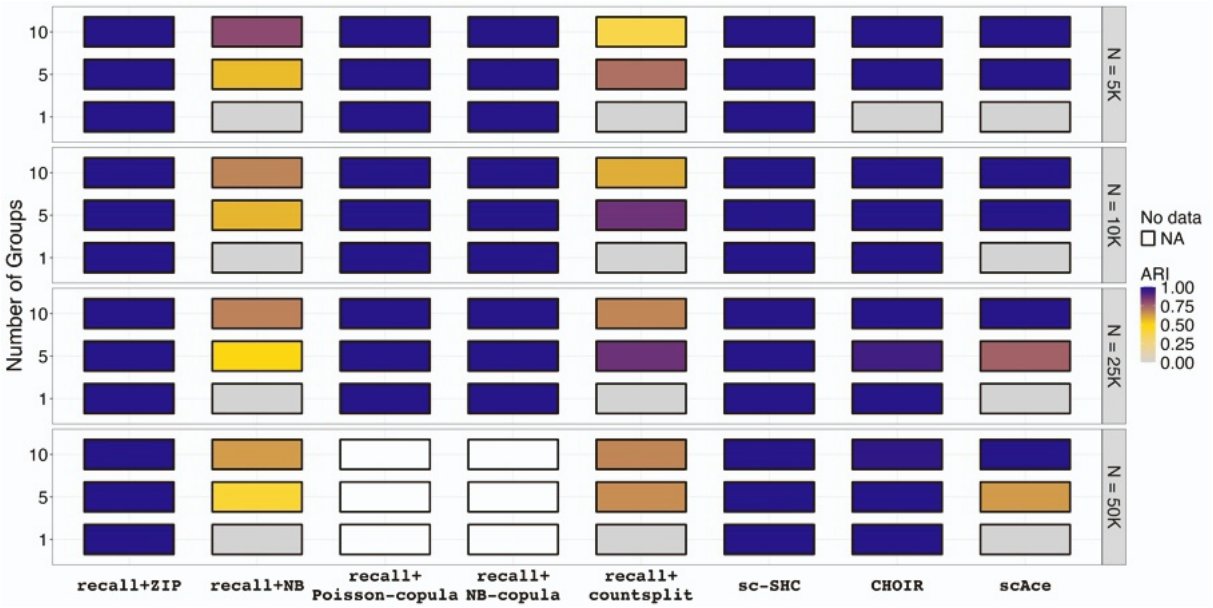


Figure S3. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets using the adjusted Rand index (ARI). Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of N = 5K, 10K, 25K, and 50K cells. The two `recall` copula algorithms scaled exponentially with the number of cells and were not able to be completed when N = 50K cells.

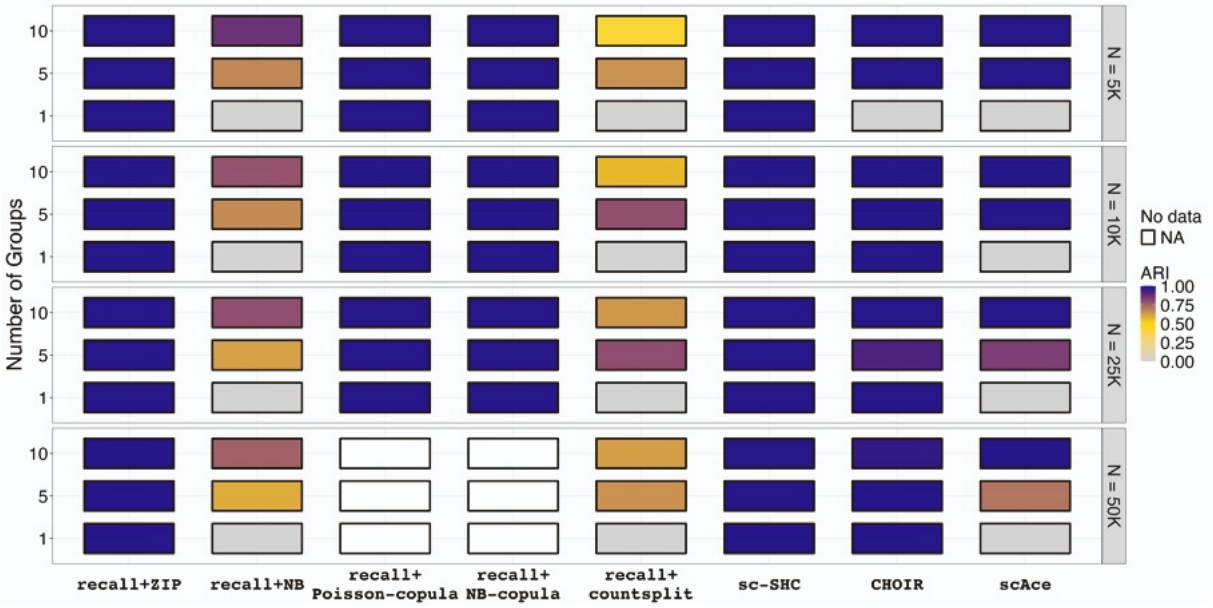


Figure S4. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets using completeness. Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of $N = 5$K, 10K, 25K, and 50K cells. The two `recall` copula algorithms scaled exponentially with the number of cells and were not able to be completed when $N = 50$K cells.

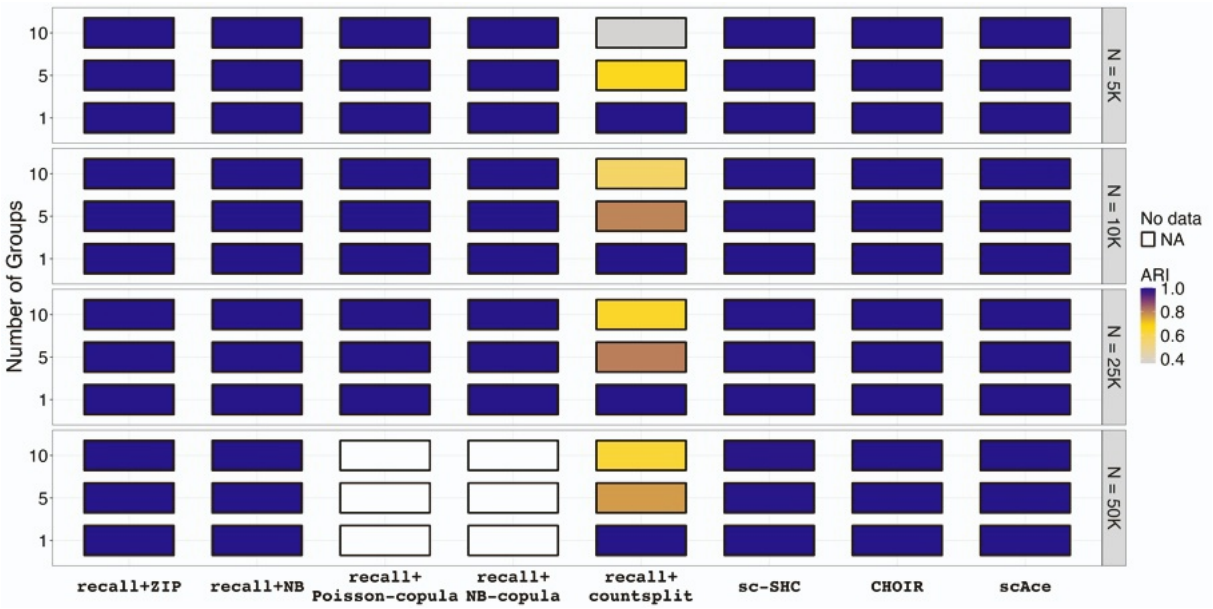


Figure S5. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets using homogeneity. Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of $N = 5$K, 10K, 25K, and 50K cells. The two `recall` copula algorithms scaled exponentially with the number of cells and were not able to be completed when $N = 50$K cells.

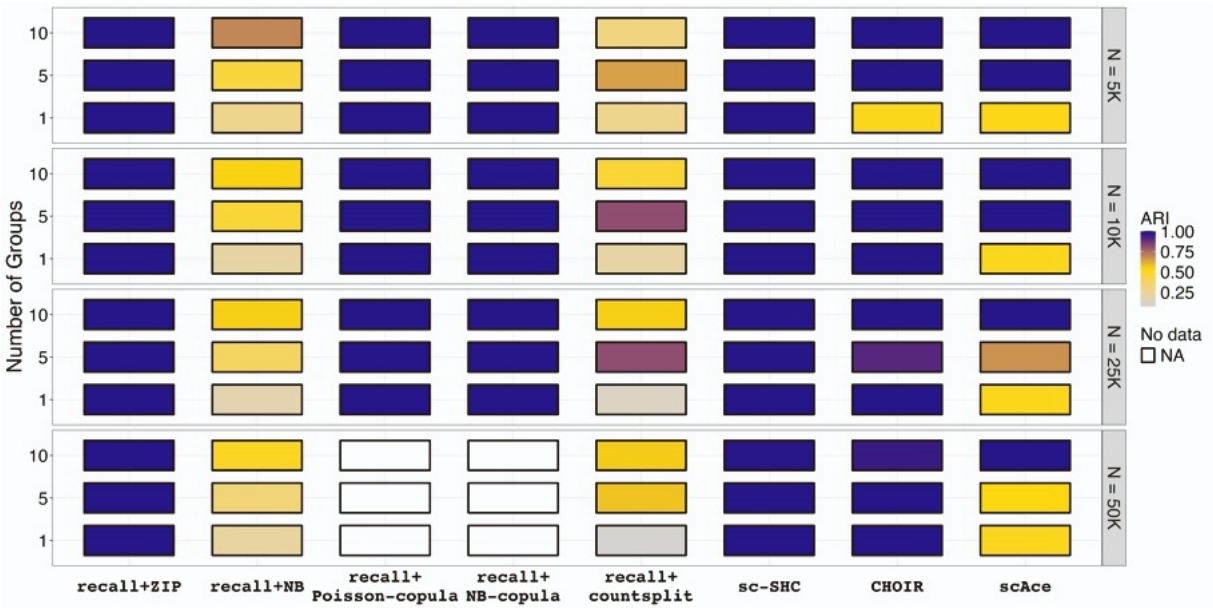


Figure S6. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets using the Jaccard index. Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of $N = 5$K, 10K, 25K, and 50K cells. The two `recall` copula algorithms scaled exponentially with the number of cells and were not able to be completed when $N =$ 50K cells.

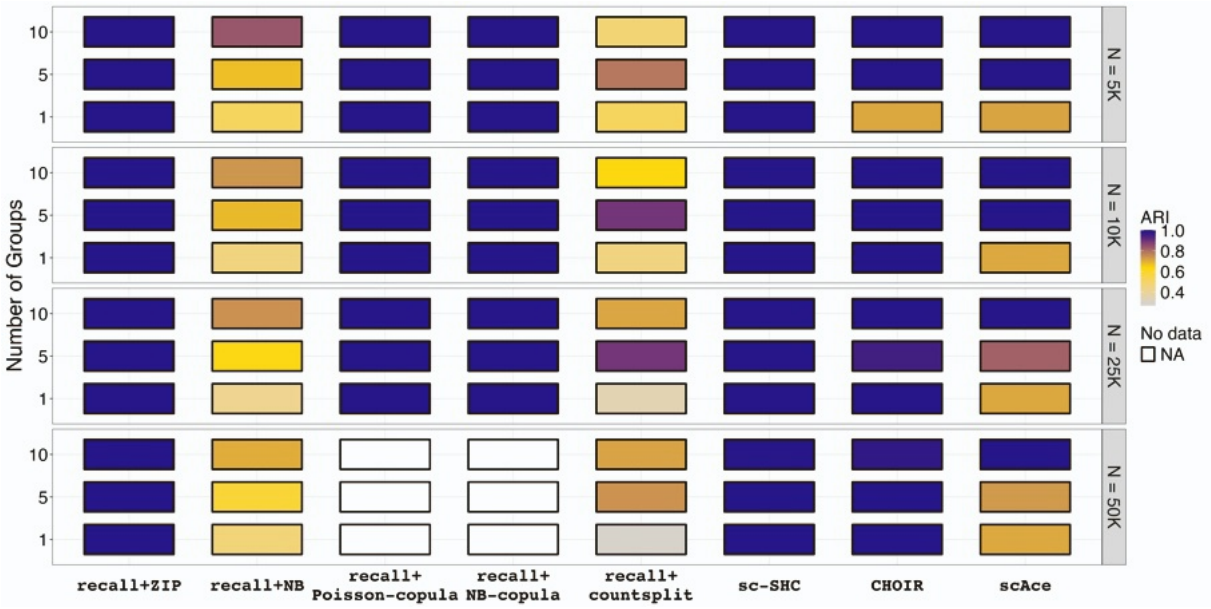


Figure S7. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets using the Fowlkes-Mallows index (FMI). Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of $N = 5$K, 10K, 25K, and 50K cells. The two `recall` copula algorithms scaled exponentially with the number of cells and were not able to be completed when $N = 50$K cells.

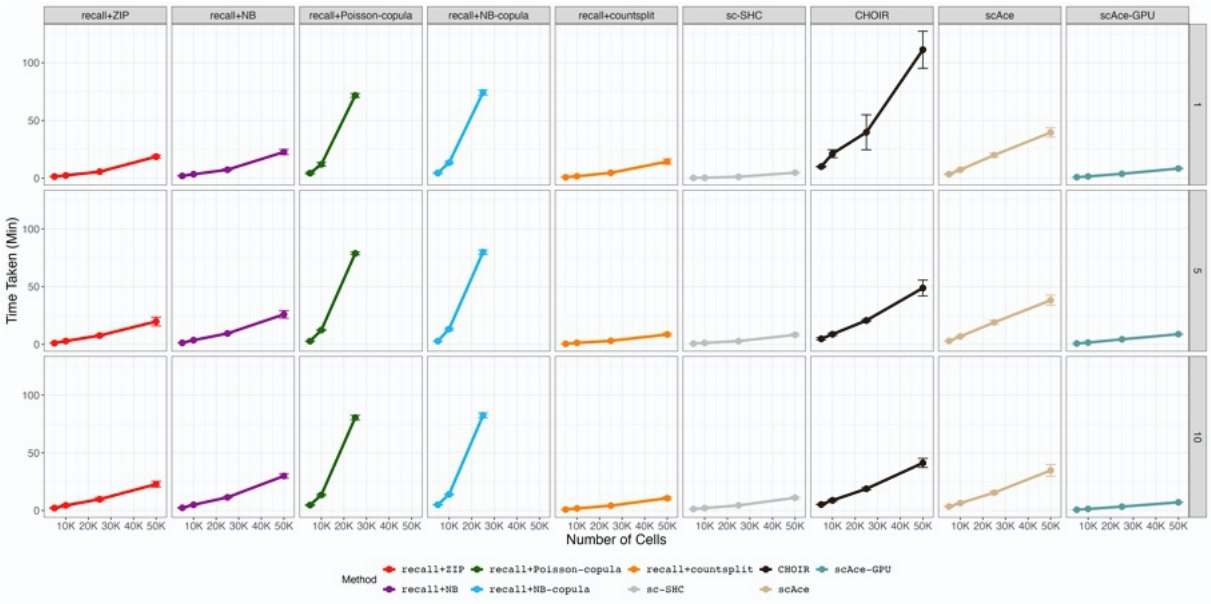


Figure S8. Comparison of runtimes (in minutes) for `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets. Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of $N = 5$K, 10K, 25K, and 50K cells. The two `recall` copula algorithms scaled exponentially with the number of cells and were not able to be completed when $N = 50$K cells.

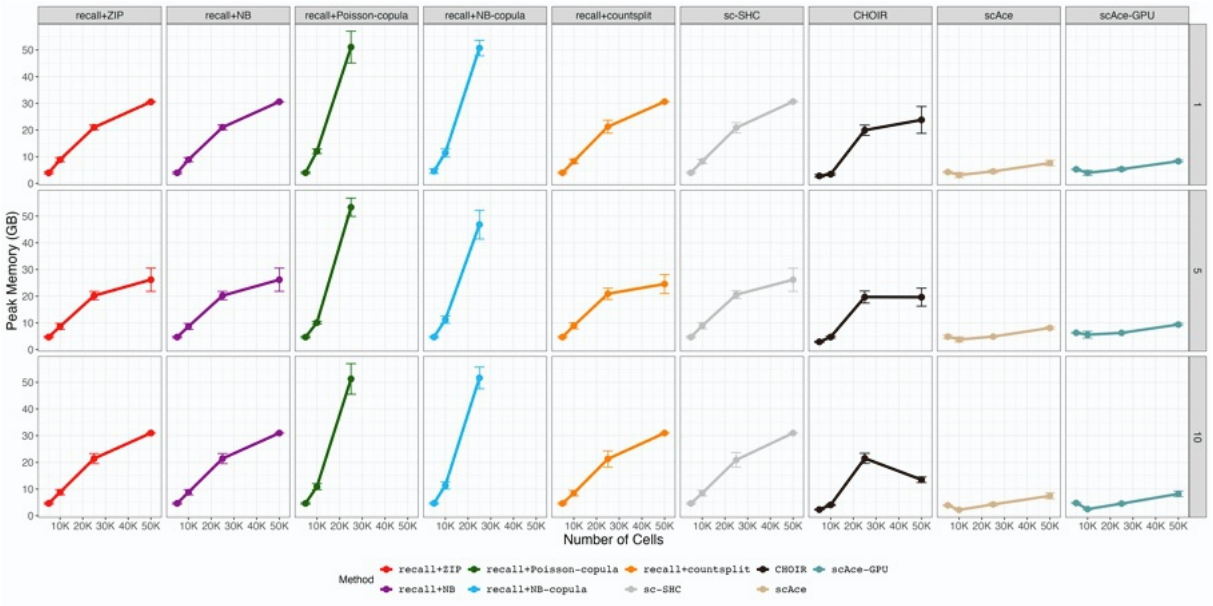


Figure S9. Comparison of peak memory usage (in gigabytes) for `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets. Each simulated dataset had five replicates and consisted of 1, 5, and 10 groups with varying sample sizes of $N = 5$K, 10K, 25K, and 50K cells. The two `recall` copula algorithms scaled exponentially with the number of cells and were not able to be completed when $N = 50$K cells.

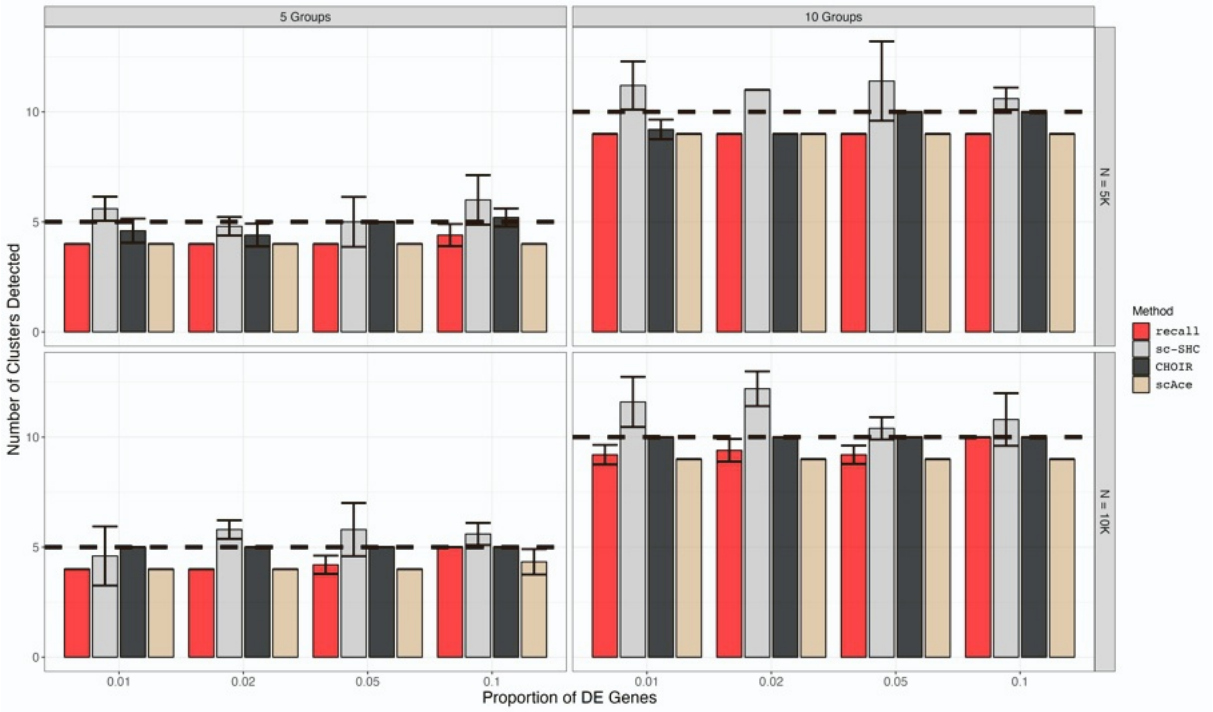


Figure S10. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets with rare cell types evaluated by the number of clusters detected. Each simulated dataset had five replicates and consisted of five or ten cell types with varying sample sizes of $N = 5$K and 10K cells. For each simulation, one cell type was chosen to be the rare cell type and was allocated approximately 1% of all simulated cells. Each remaining cell type was sampled with approximately equal proportions. The rare cell type was then simulated with 0.01, 0.02, 0.05, and 0.1 as its proportion of differentially expressed genes, while the remaining cell types were simulated with 0.1 as their proportion of differentially expressed genes. The correct number of cells is denoted by the dashed line. Perfect performance is when a given bar is touching the dashed line. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the rare cell type.

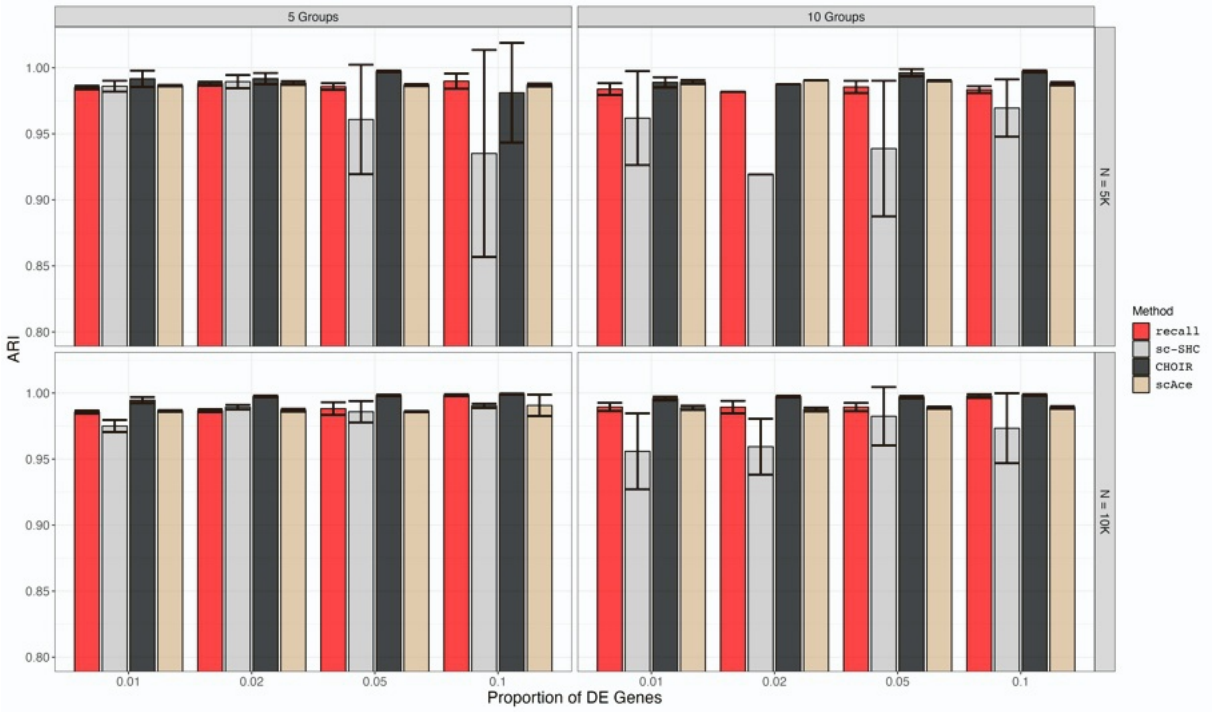


Figure S11. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets with rare cell types evaluated by adjusted Rand index (ARI). Each simulated dataset had five replicates and consisted of five or ten cell types with varying sample sizes of $N = 5$K and 10K cells. For each simulation, one cell type was chosen to be the rare cell type and was allocated approximately 1% of all simulated cells. Each remaining cell type was sampled with approximately equal proportions. The rare cell type was then simulated with 0.01, 0.02, 0.05, and 0.1 as its proportion of differentially expressed genes, while the remaining cell types were simulated with 0.1 as their proportion of differentially expressed genes. The ARI is shown on the y-axis and perfect performance is 1.0. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the rare cell type.

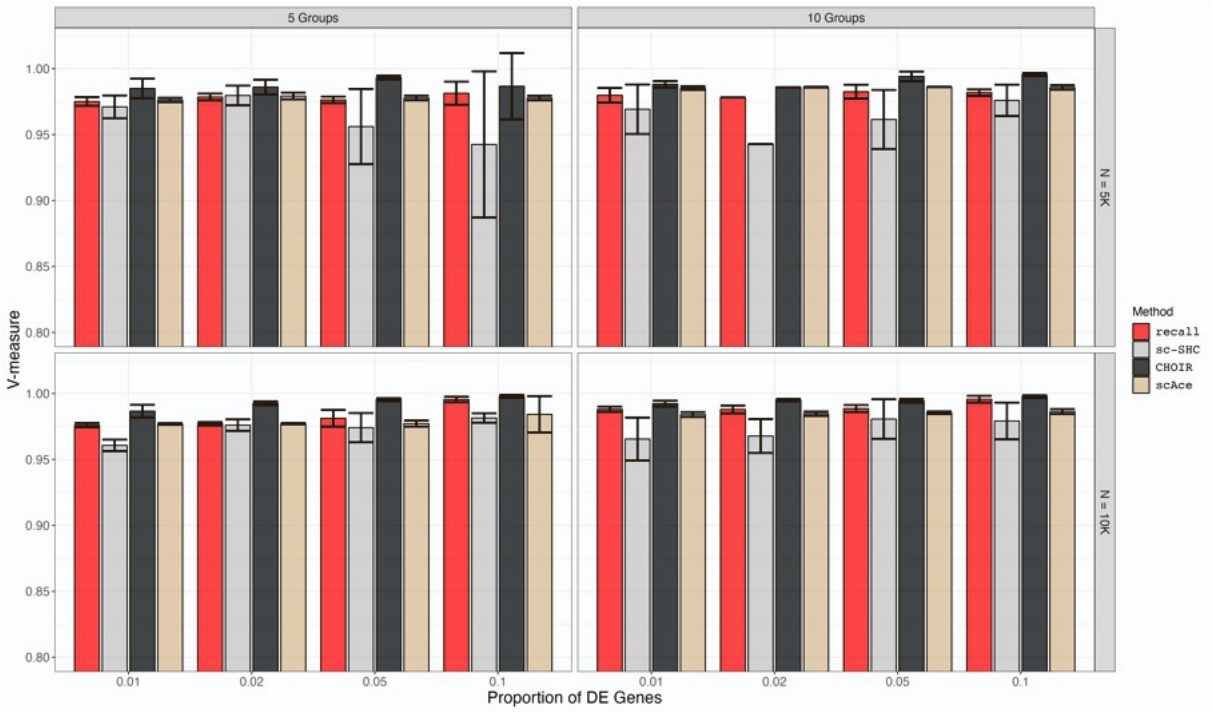


Figure S12. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets with rare cell types evaluated by V-measure. Each simulated dataset had five replicates and consisted of five or ten cell types with varying sample sizes of N = 5K and 10K cells. For each simulation, one cell type was chosen to be the rare cell type and was allocated approximately 1% of all simulated cells. Each remaining cell type was sampled with approximately equal proportions. The rare cell type was then simulated with 0.01, 0.02, 0.05, and 0.1 as its proportion of differentially expressed genes, while the remaining cell types were simulated with 0.1 as their proportion of differentially expressed genes. The V-measure is shown on the y-axis and perfect performance is 1.0. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the rare cell type.

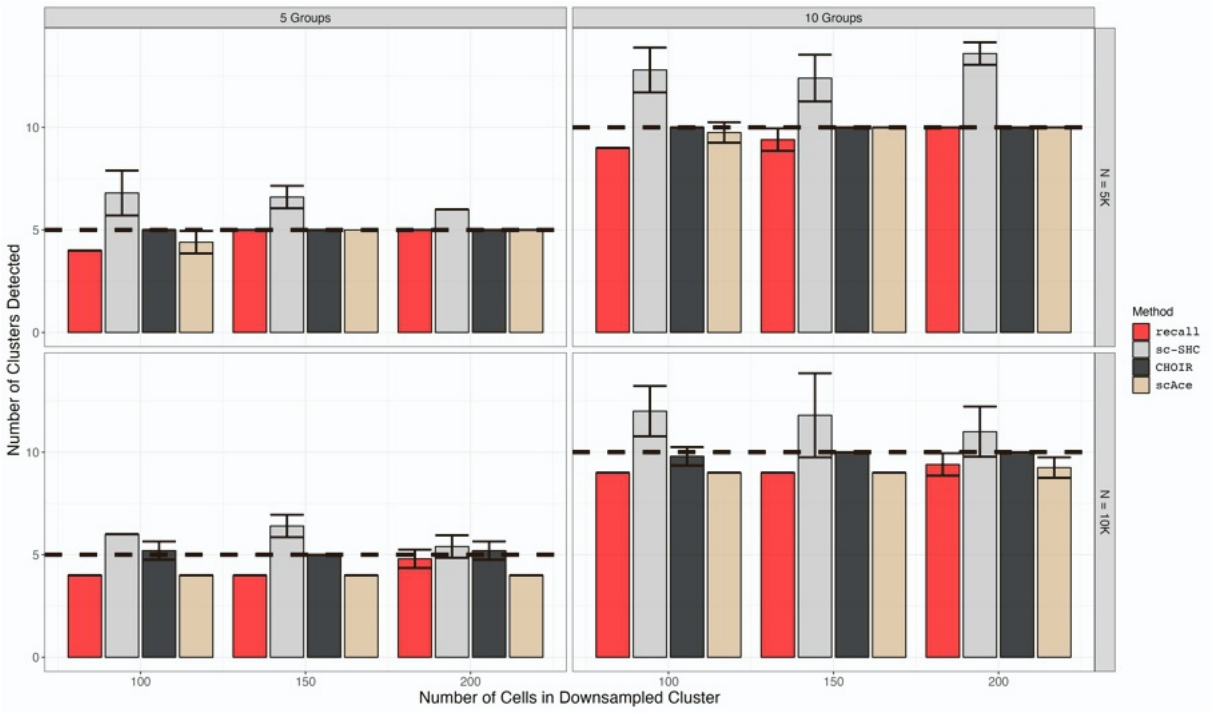


Figure S13. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets with rare cell types evaluated by the number of clusters detected. Each simulated dataset had five replicates and consisted of 5 or 10 cell types with varying sample sizes of $N = 5$K and 10K cells. Each cell type was sampled with approximately equal proportions. For each simulation, one cell type was chosen to be the rare cell type. This cell type was then downsampled to 100, 150, and 200 cells (unless that involved going beyond the original number of cells for that cell type). The correct number of cells is denoted by the dashed line. Perfect performance is when a given bar is touching the dashed line. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the rare cell type.

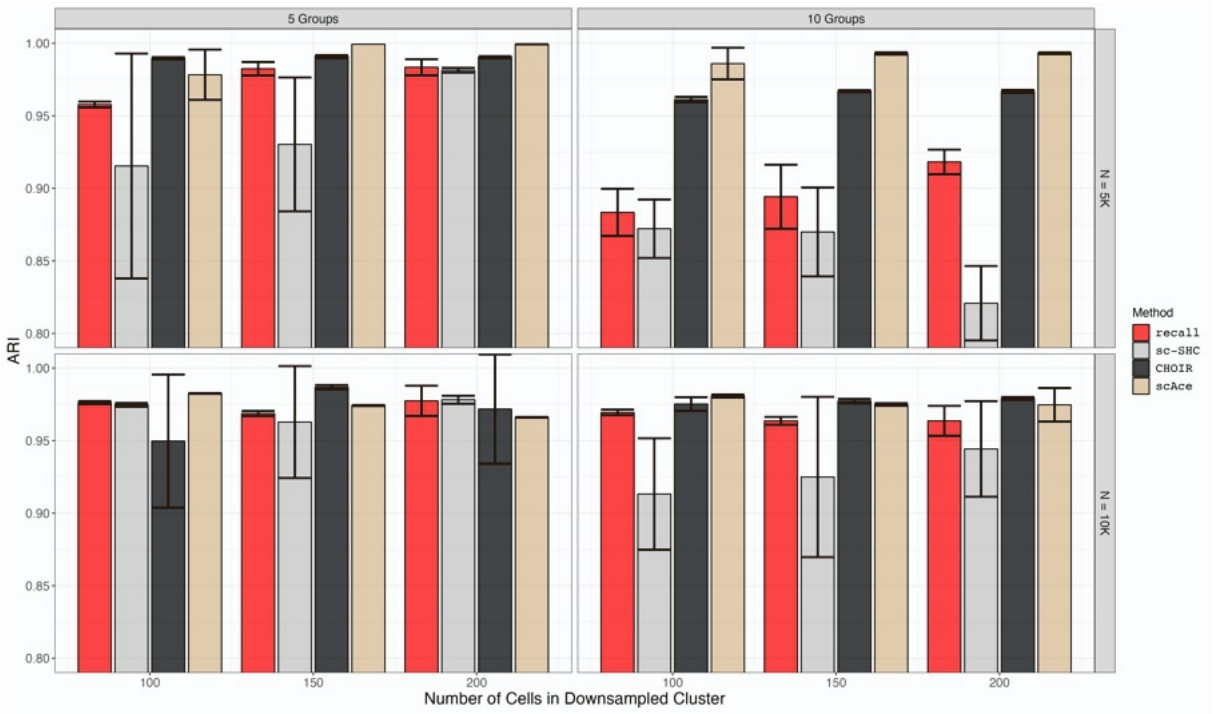


Figure S14. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets with rare cell types evaluated by adjusted Rand index (ARI). Each simulated dataset had five replicates and consisted of 5 or 10 cell types with varying sample sizes of N = 5K and 10K cells. Each cell type was sampled with approximately equal proportions. For each simulation, one cell type was chosen to be the rare cell type. This cell type was then downsampled to 100, 150, and 200 cells (unless that involved going beyond the original number of cells for that cell type). The ARI is shown on the y-axis and perfect performance is 1.0. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the rare cell type.

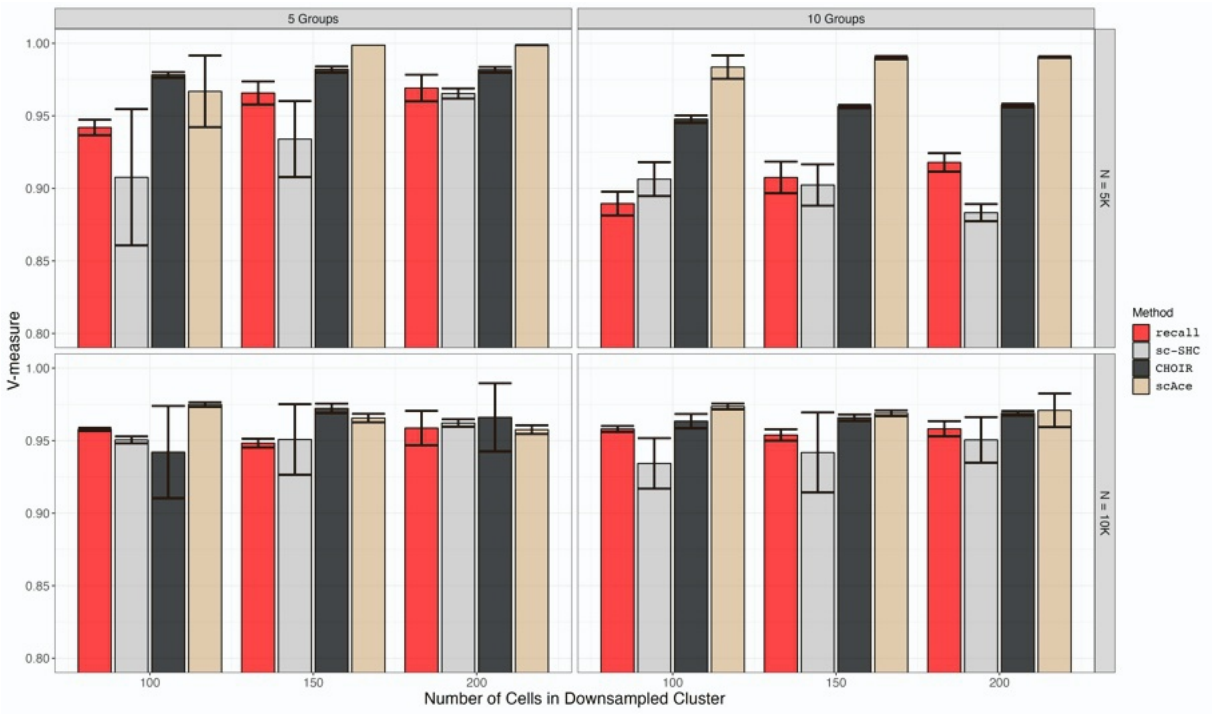


Figure S15. Performance comparison of `recall`, `sc-SHC`, `CHOIR`, and `scAce` on simulated datasets with rare cell types evaluated by V-measure. Each simulated dataset had five replicates and consisted of 5 or 10 cell types with varying sample sizes of $N = 5$K and 10K cells. Each cell type was sampled with approximately equal proportions. For each simulation, one cell type was chosen to be the rare cell type. This cell type was then downsampled to 100, 150, and 200 cells (unless that involved going beyond the original number of cells for that cell type). The V-measure is shown on the y-axis and perfect performance is 1.0. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the rare cell type.

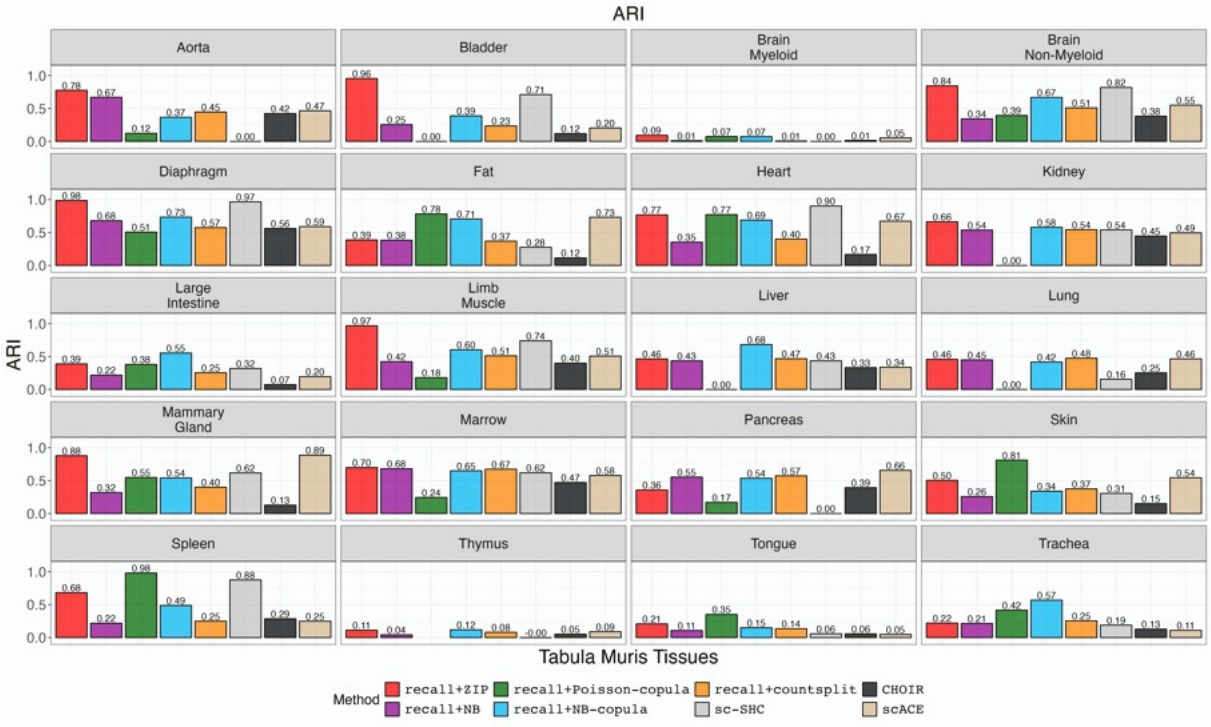


Figure S16. Performance comparison of each variant of the `recall` algorithm, `sc-SHC`, `CHOIR`, and `scAce` using the adjusted Rand index (ARI) for each tissue in the Tabula Muris dataset.

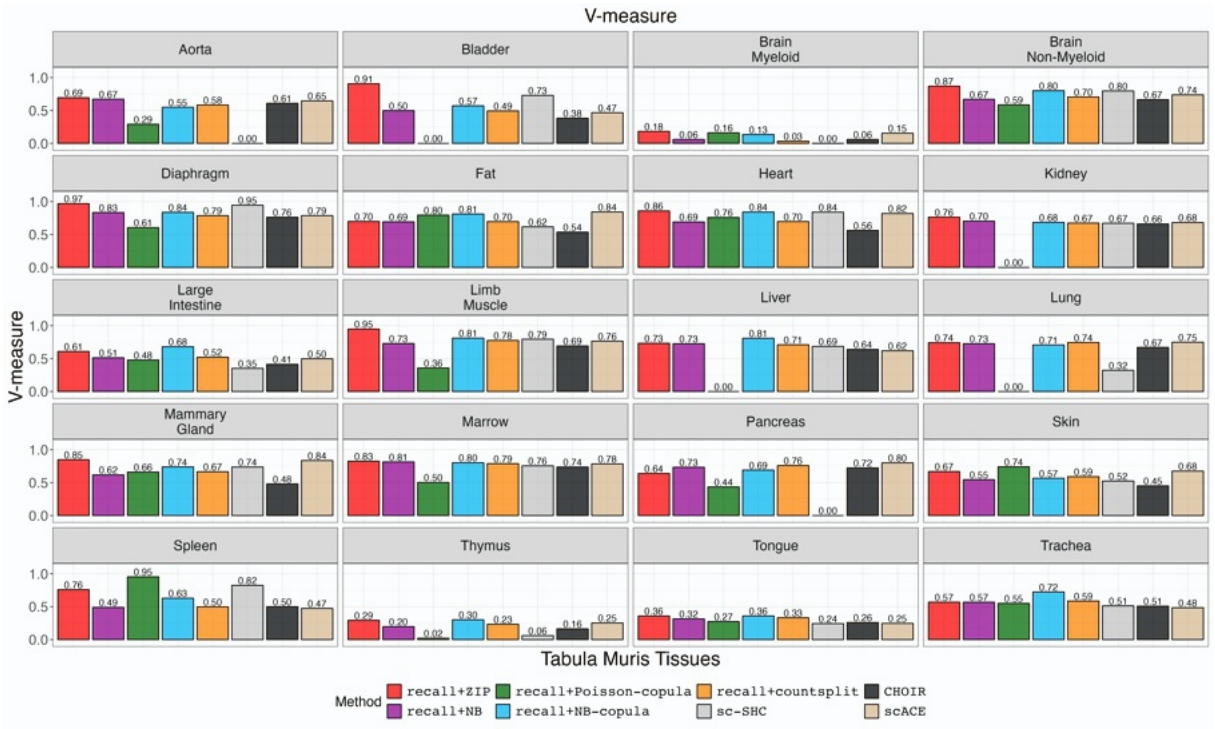


Figure S17. Performance comparison of each variant of the `recall` algorithm, `sc-SHC`, `CHOIR`, and `scAce` using V-measure for each tissue in the Tabula Muris dataset.

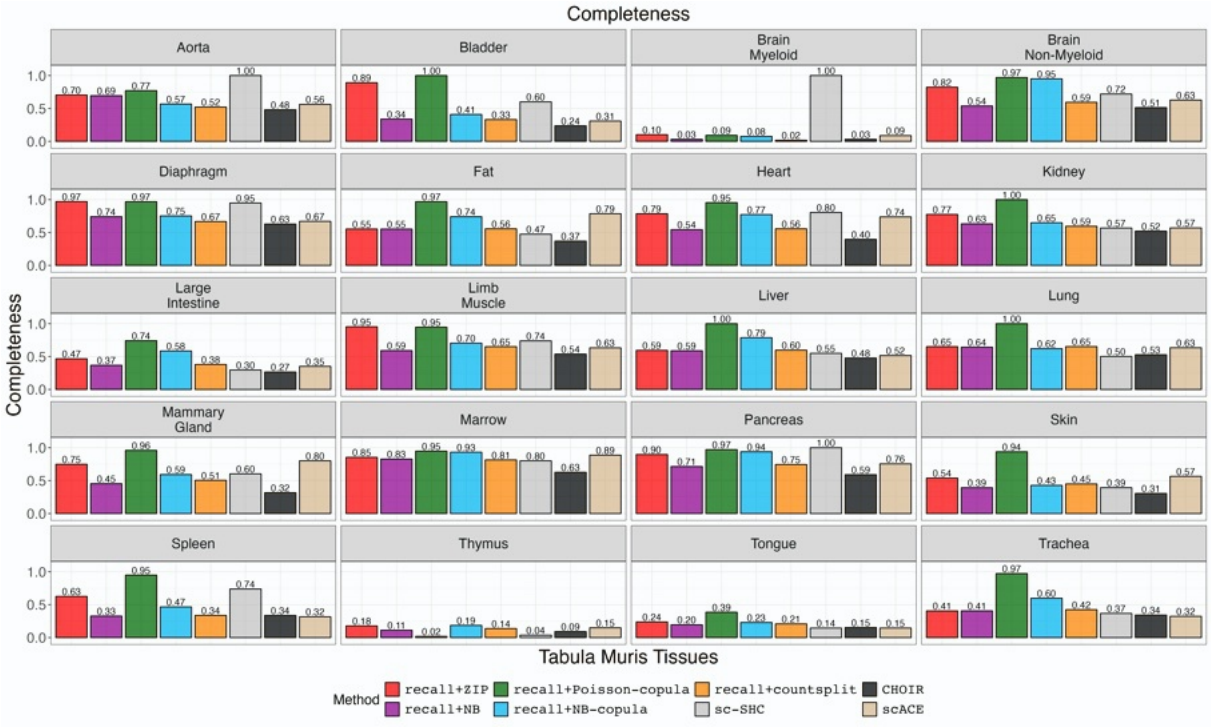


Figure S18. Performance comparison of each variant of the `recall` algorithm, `sc-SHC`, `CHOIR`, and `scAce` using completeness for each tissue in the Tabula Muris dataset.

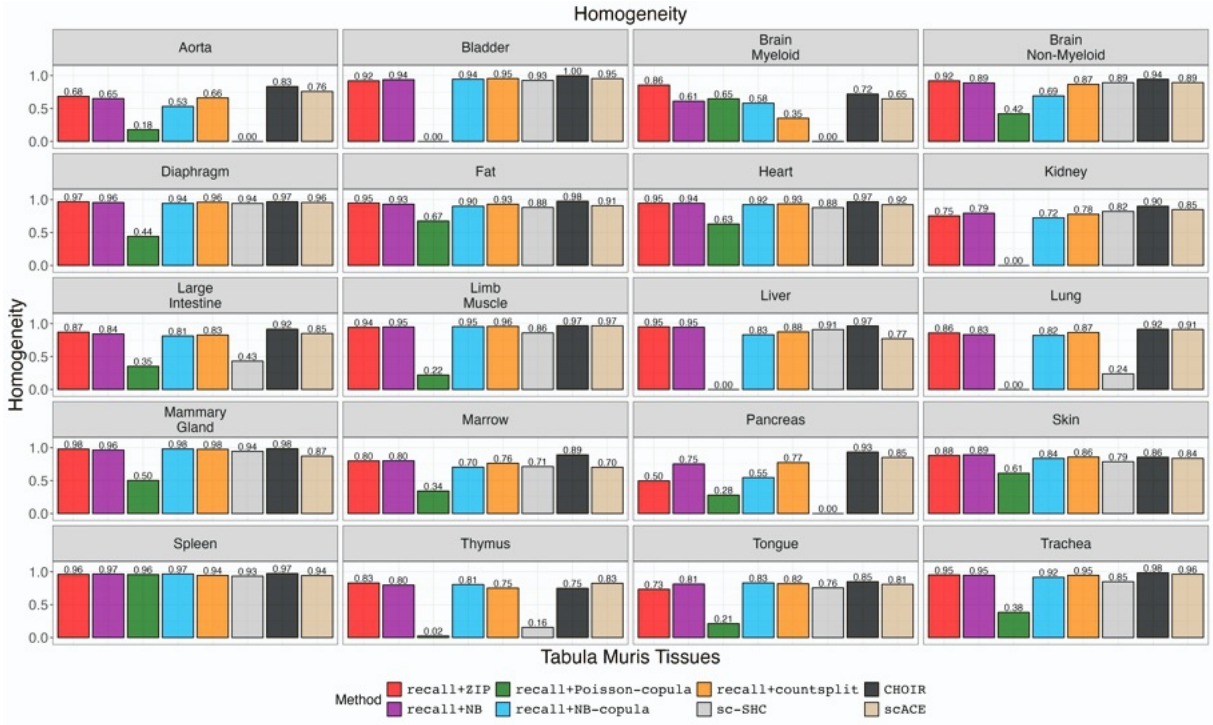


Figure S19. Performance comparison of each variant of the `recall` algorithm, `sc-SHC`, `CHOIR`, and `scAce` using homogeneity for each tissue in the Tabula Muris dataset.

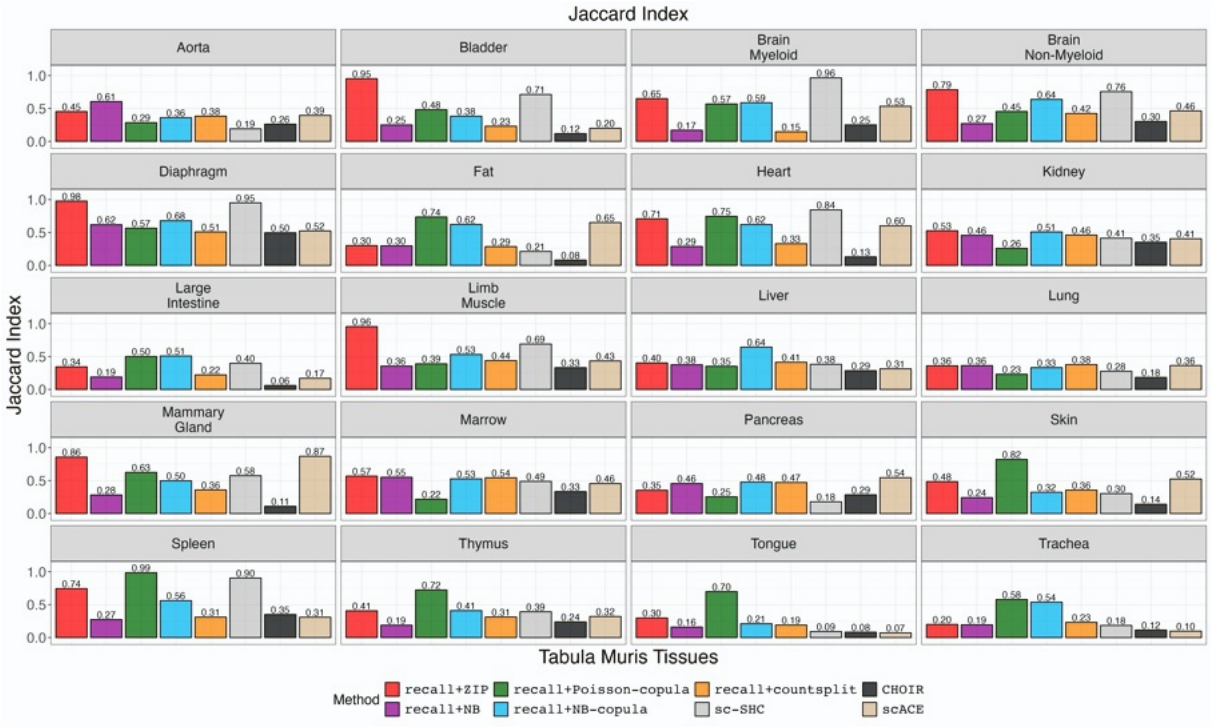


Figure S20. Performance comparison of each variant of the `recall` algorithm, `sc-SHC`, `CHOIR`, and `scAce` using the Jaccard index for each tissue in the Tabula Muris dataset.

Figure S21. Performance comparison of each variant of the `recall` algorithm, `sc-SHC`, `CHOIR`, and `scAce` using the Fowlkes-Mallows index (FMI) for each tissue in the Tabula Muris dataset.

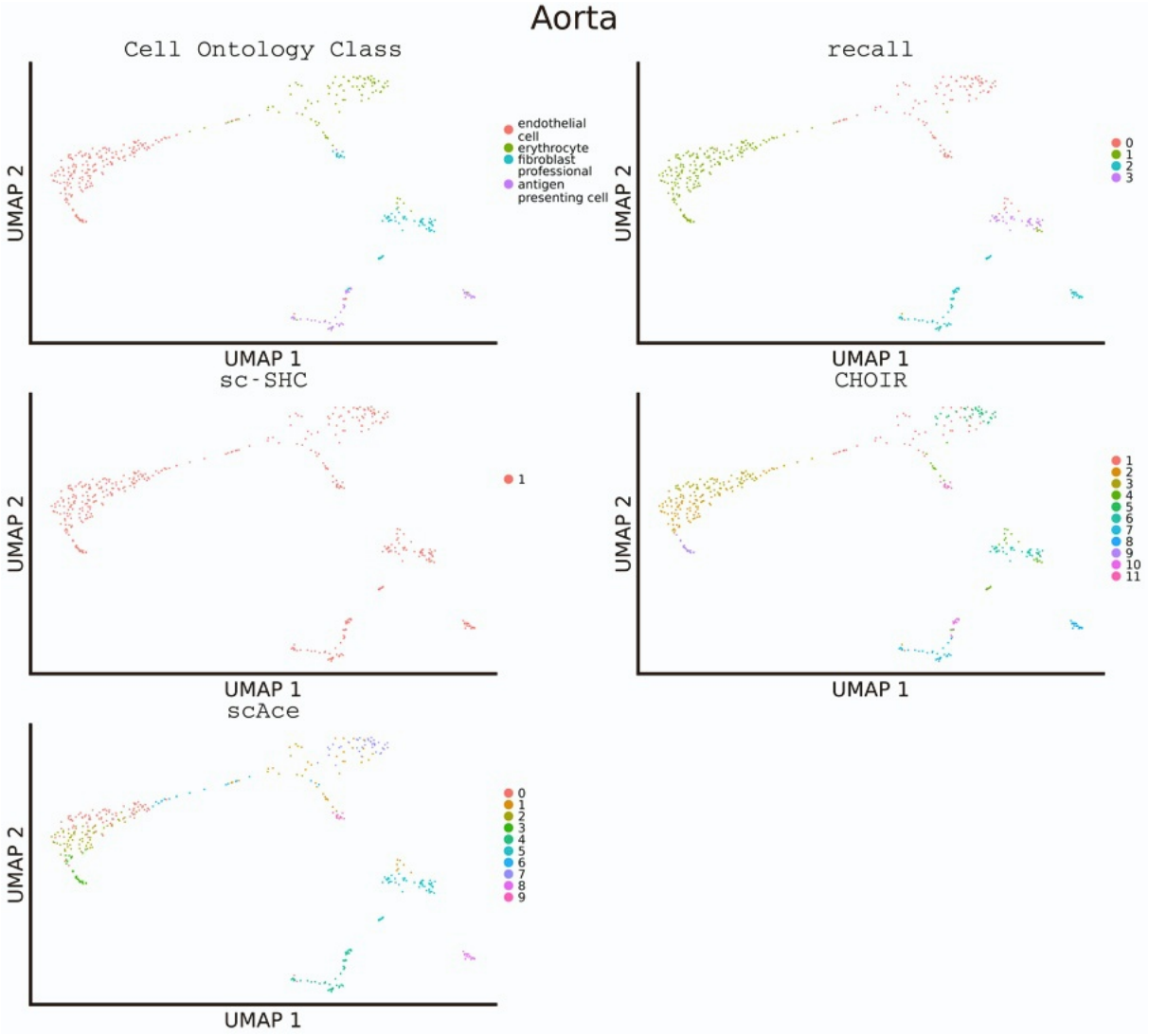


Figure S22. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the aorta tissue from the Tabula Muris study.

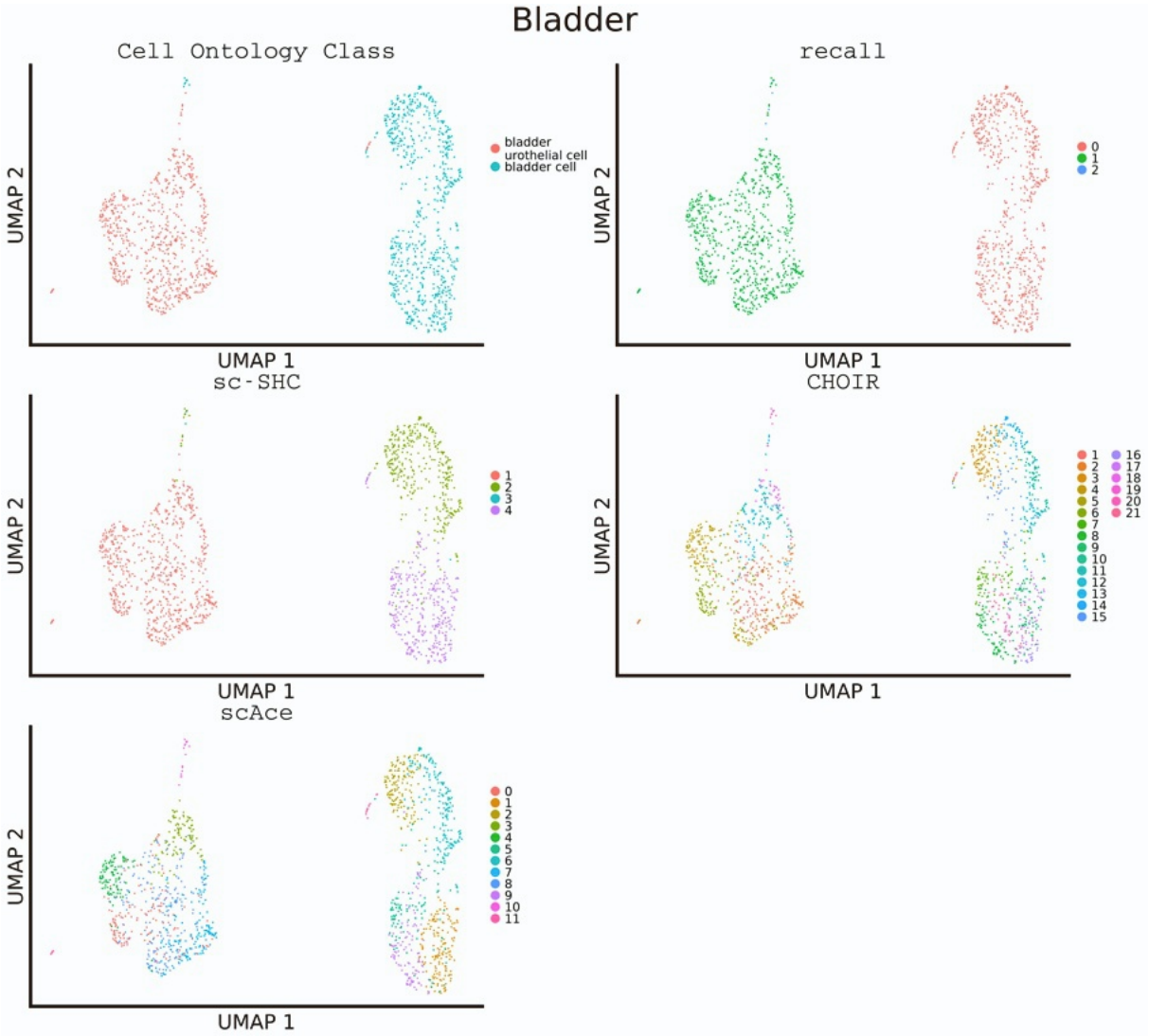


Figure S23. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the bladder tissue from the Tabula Muris study.

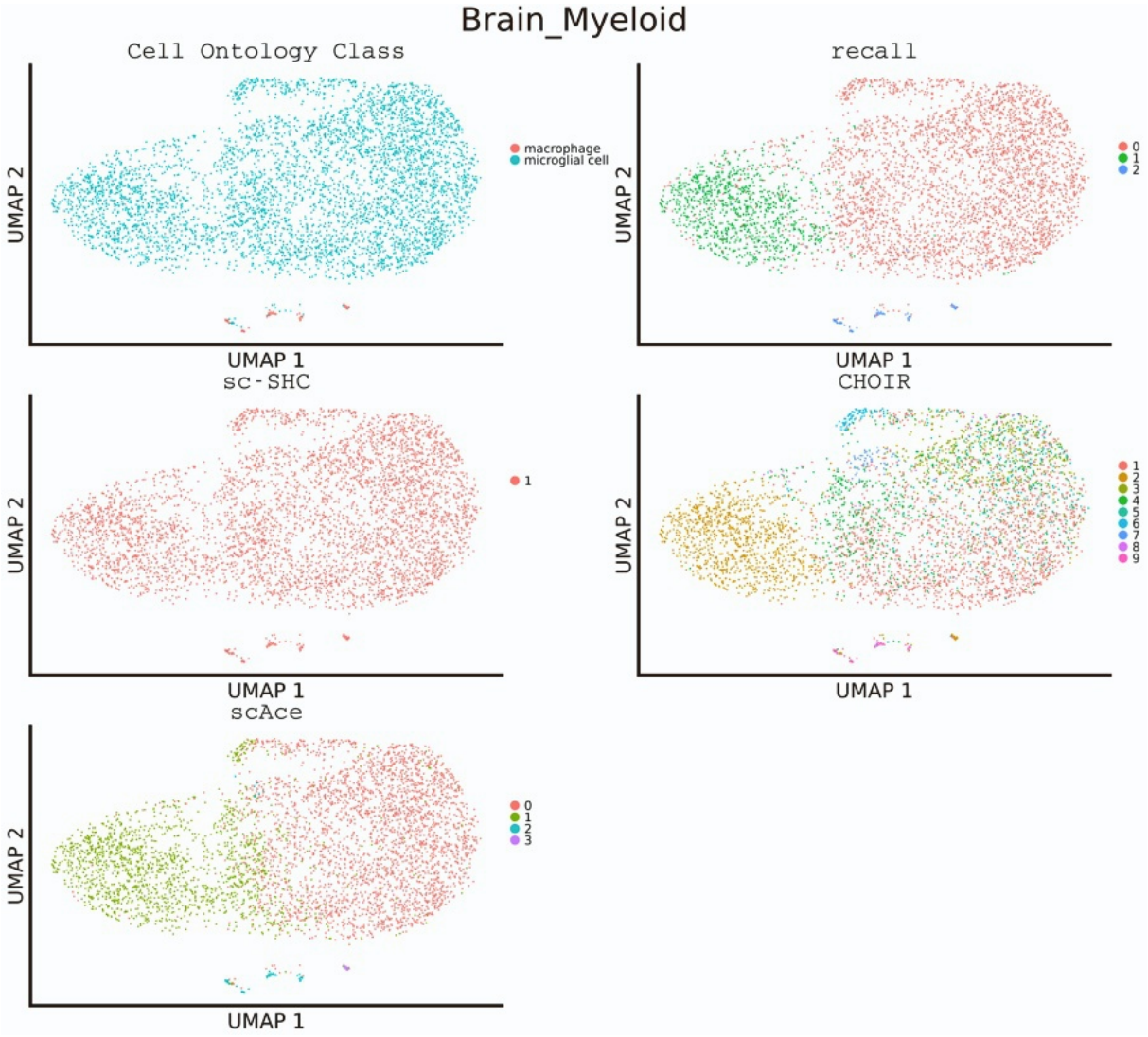


Figure S24. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the Brain myeloid tissue from the Tabula Muris study.

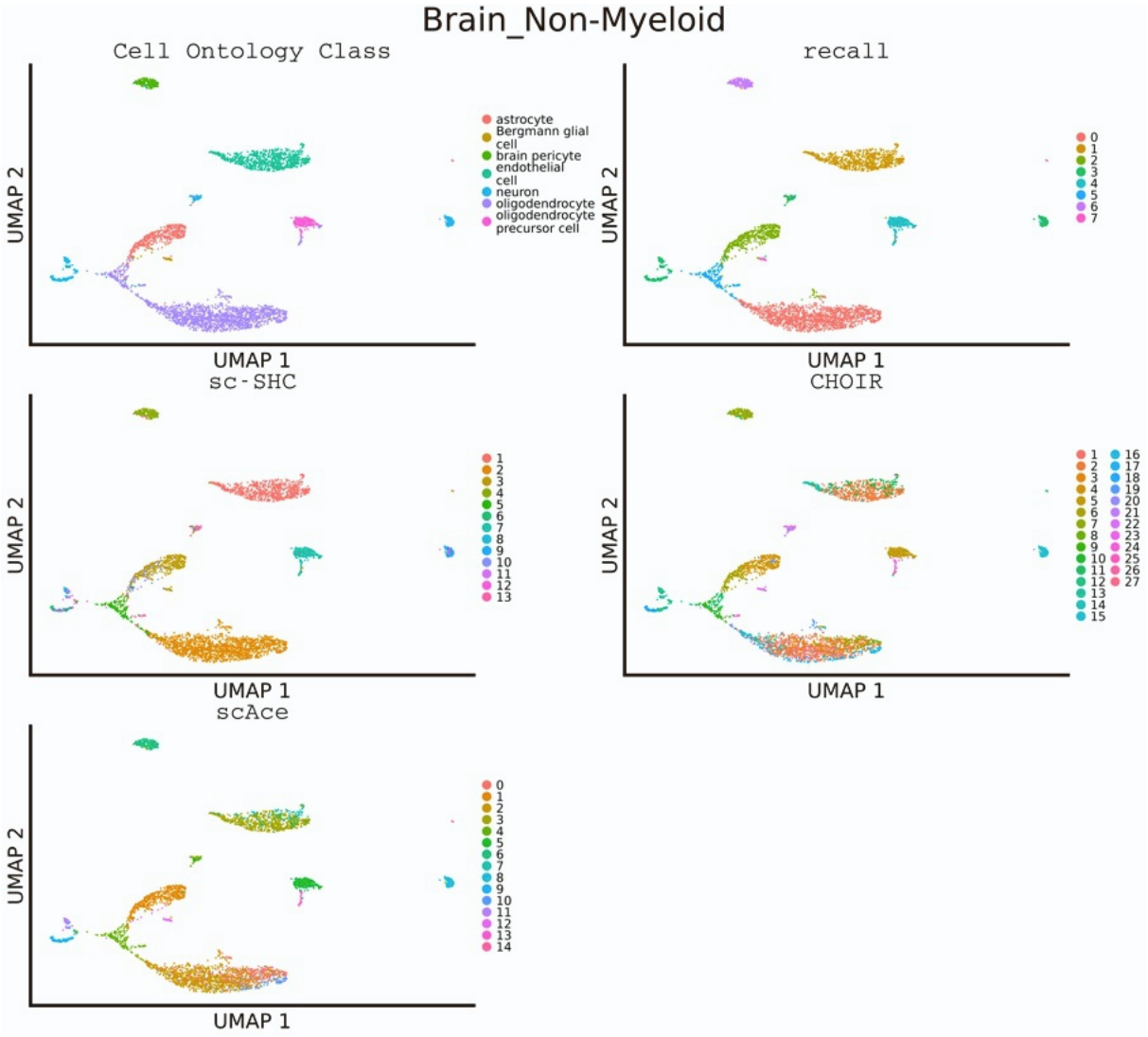


Figure S25. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the Brain non-myeloid tissue from the Tabula Muris study.

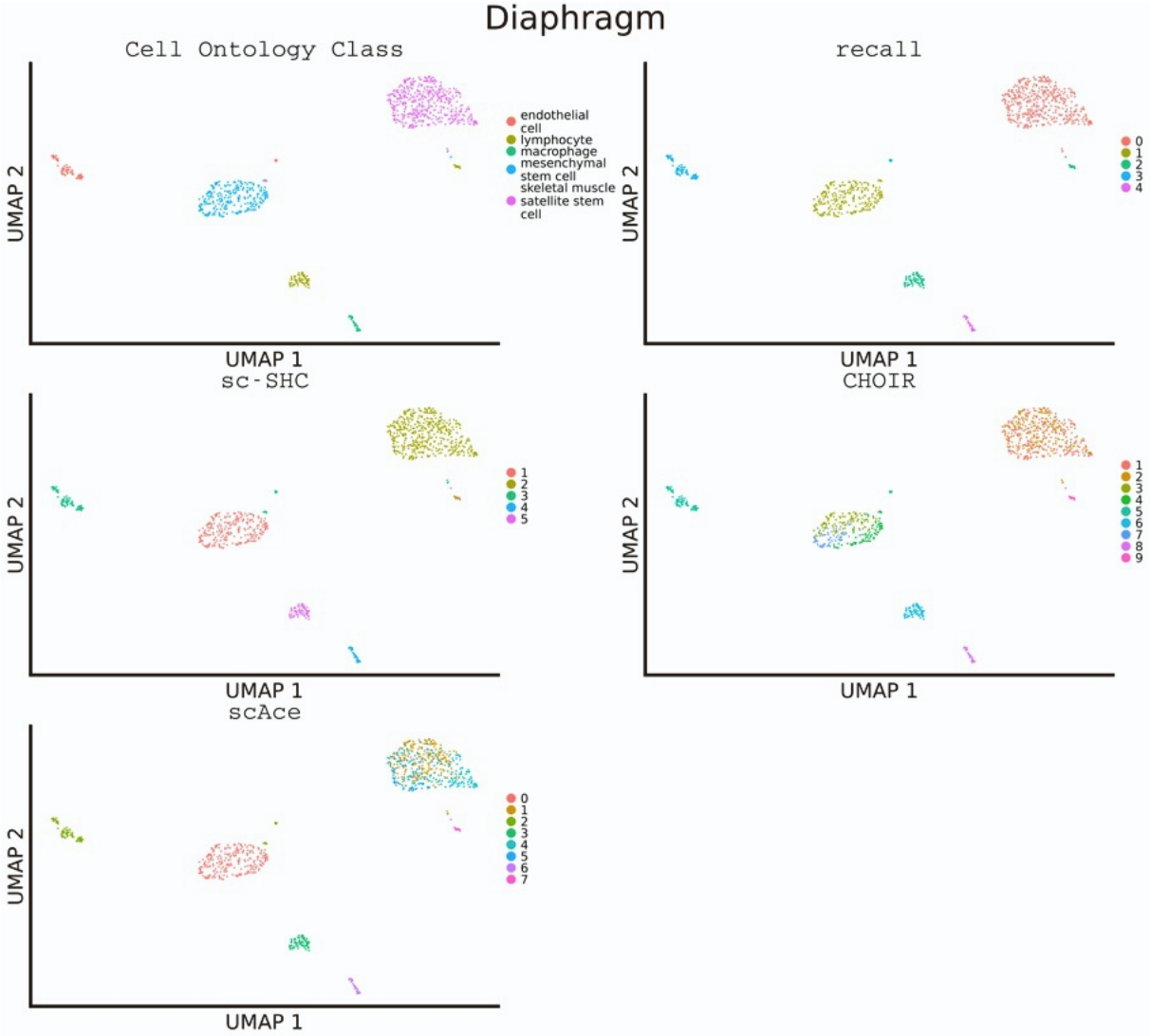


Figure S26. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the diaphragm tissue from the Tabula Muris study.

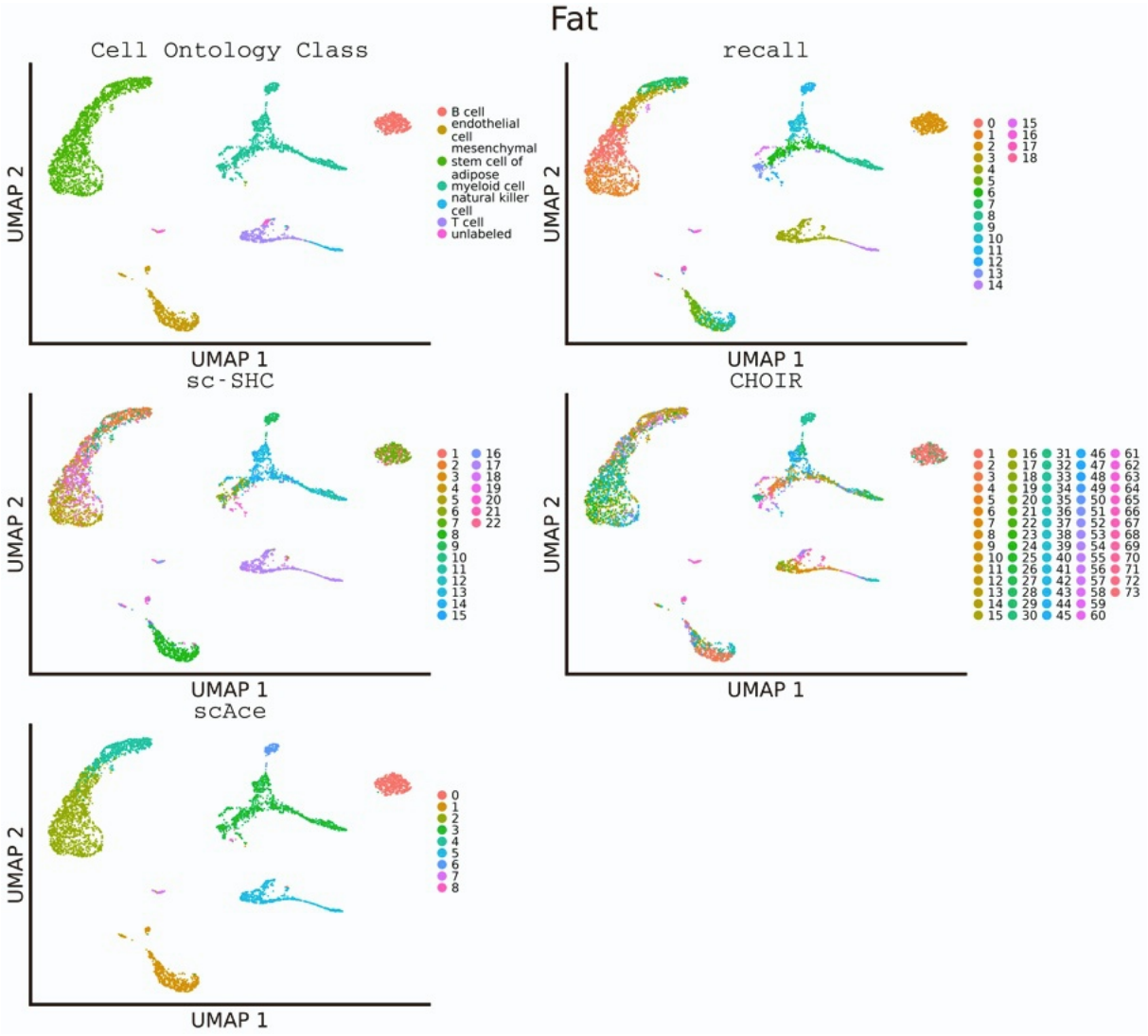


Figure S27. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the fat tissue from the Tabula Muris study.

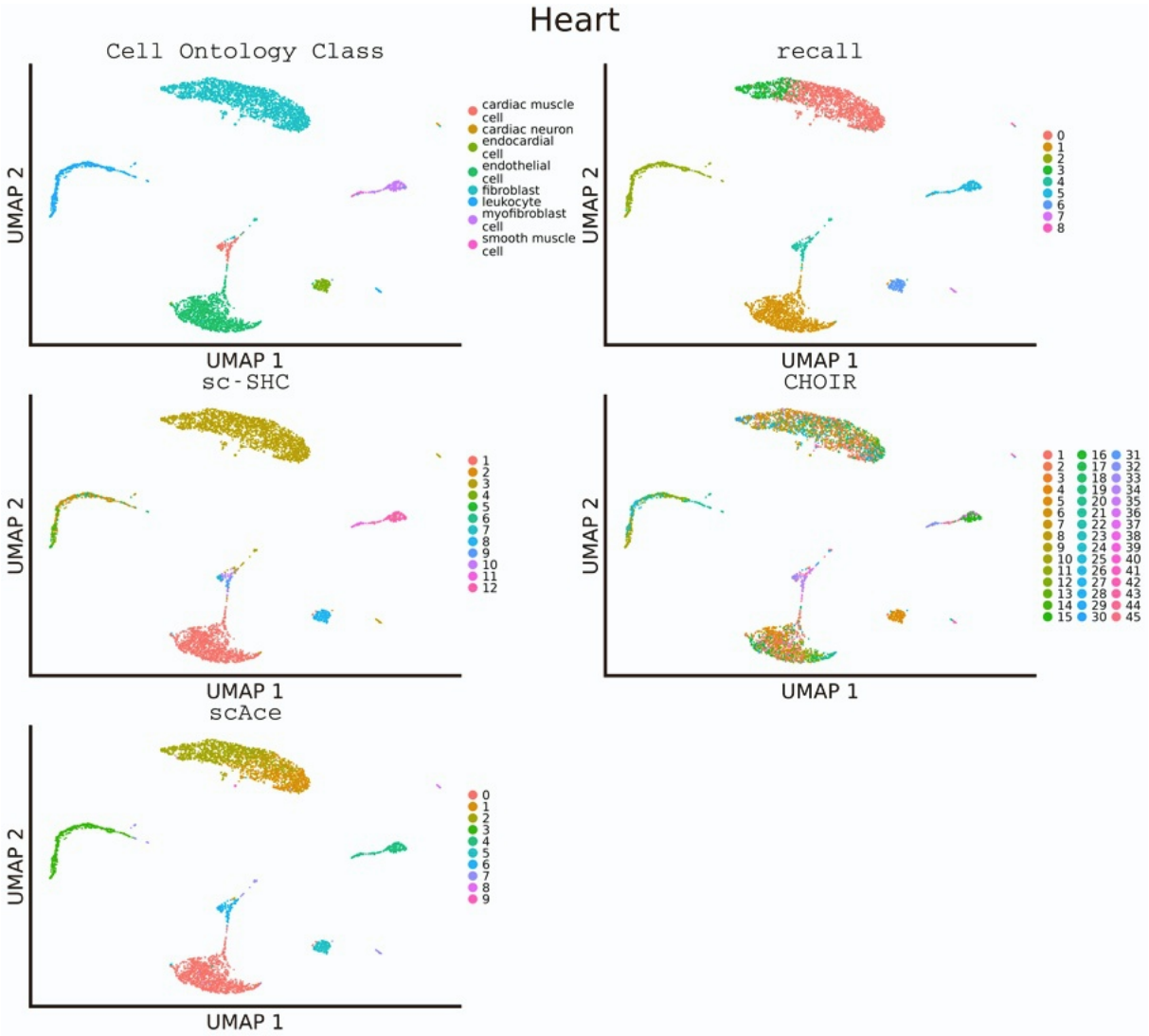


Figure S28. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the heart tissue from the Tabula Muris study.

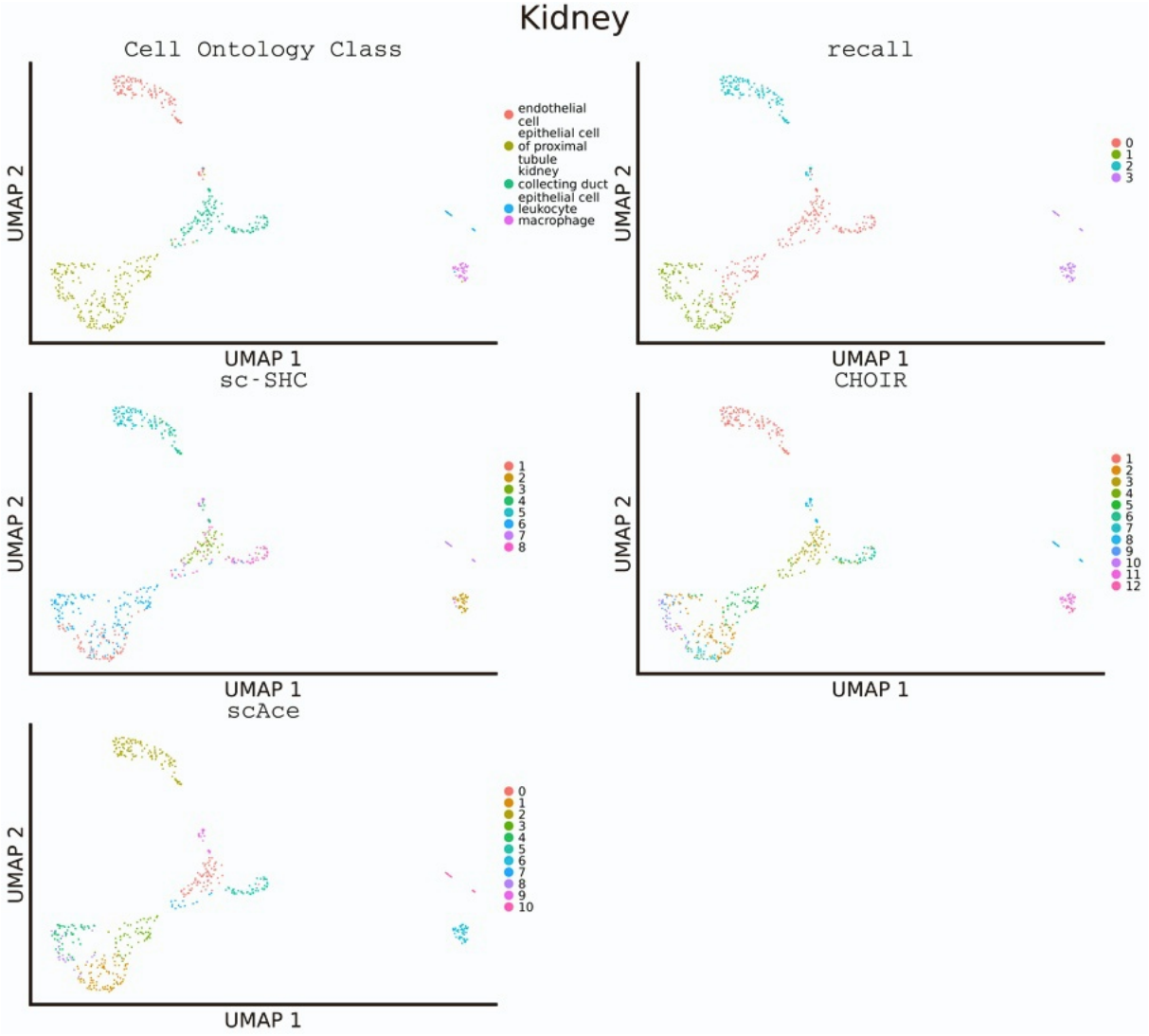


Figure S29. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the kidney tissue from the Tabula Muris study.

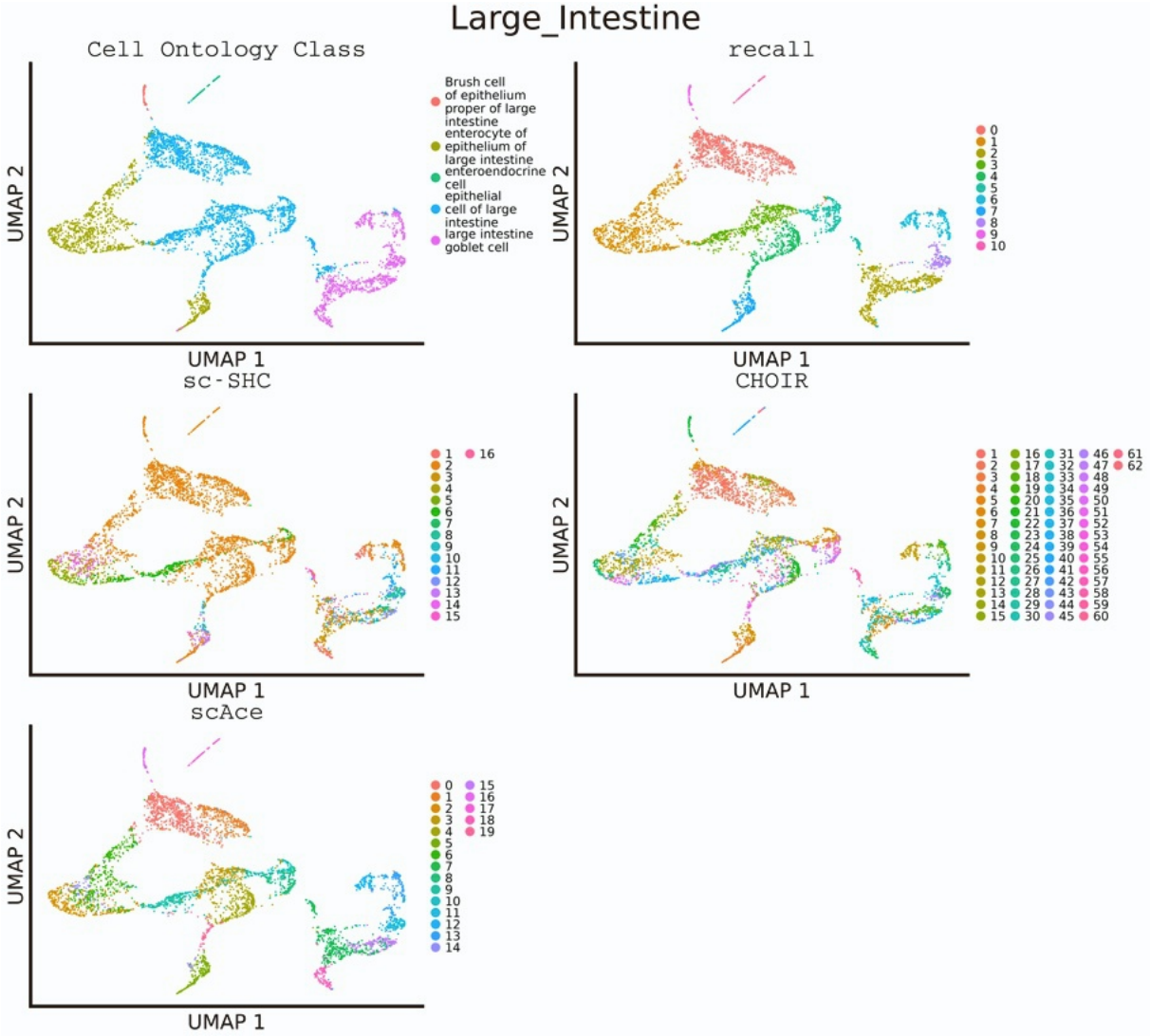


Figure S30. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the large intestine tissue from the Tabula Muris study.

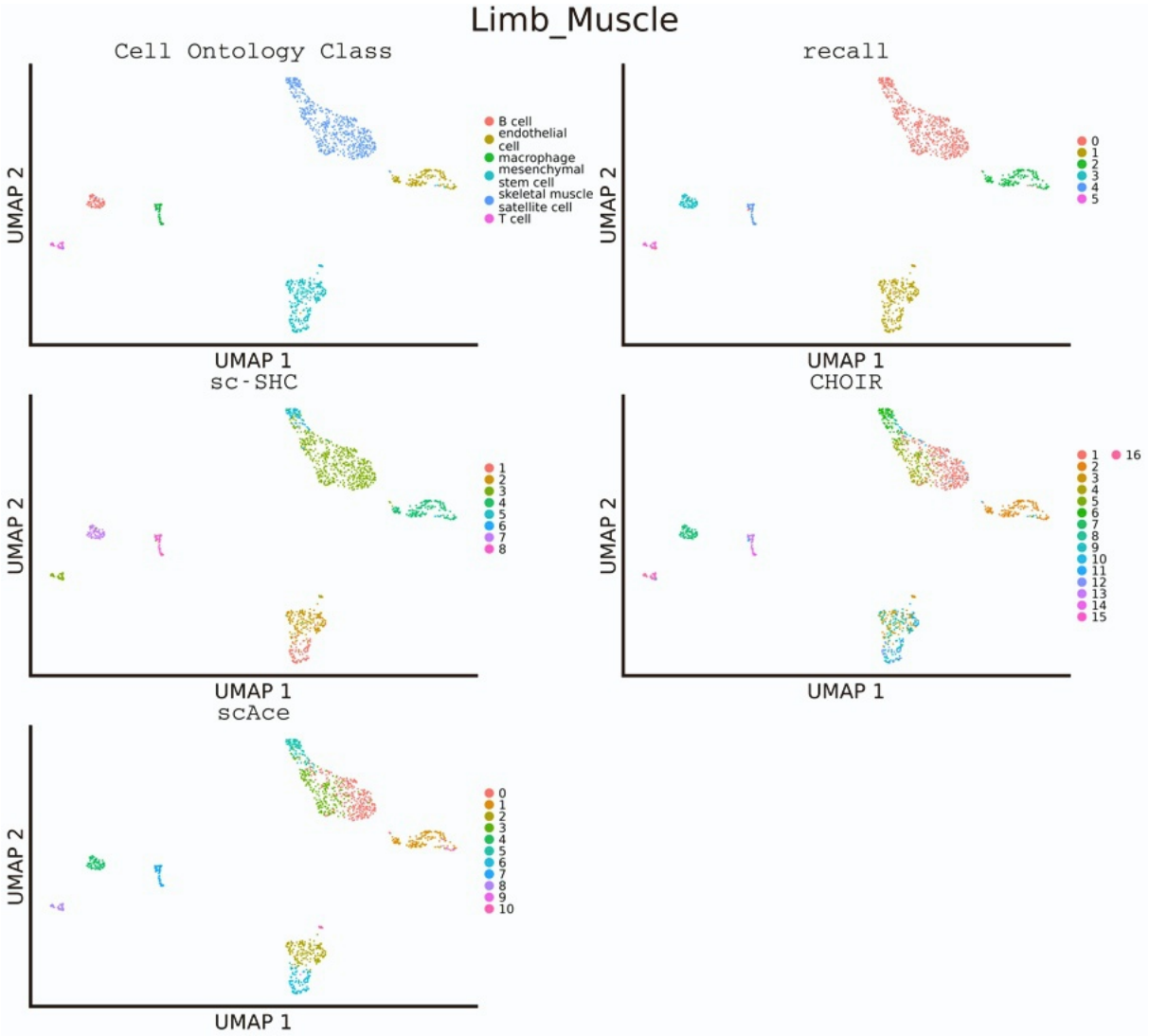


Figure S31. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the limb muscle tissue from the Tabula Muris study.

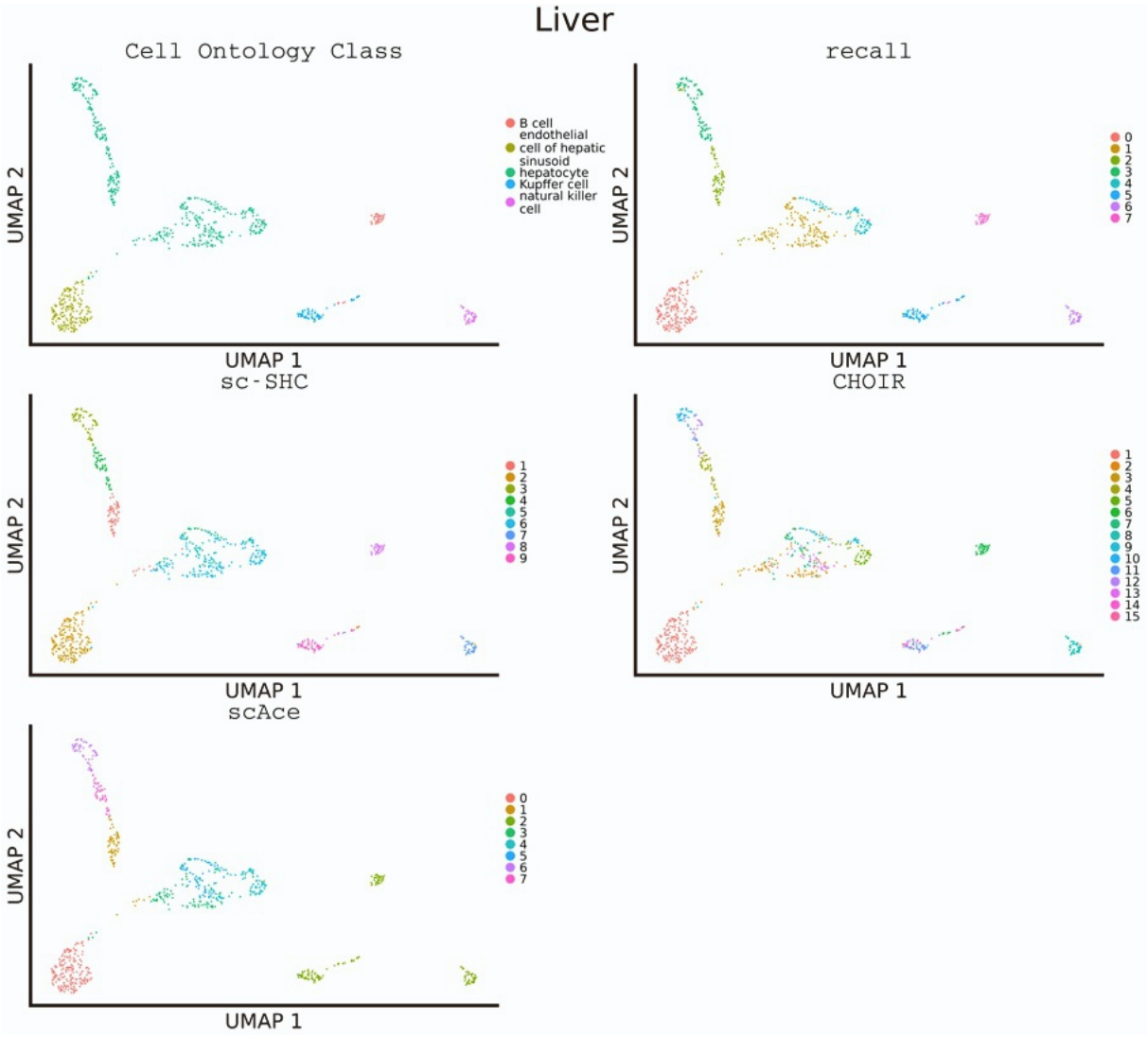


Figure S32. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the liver tissue from the Tabula Muris study.

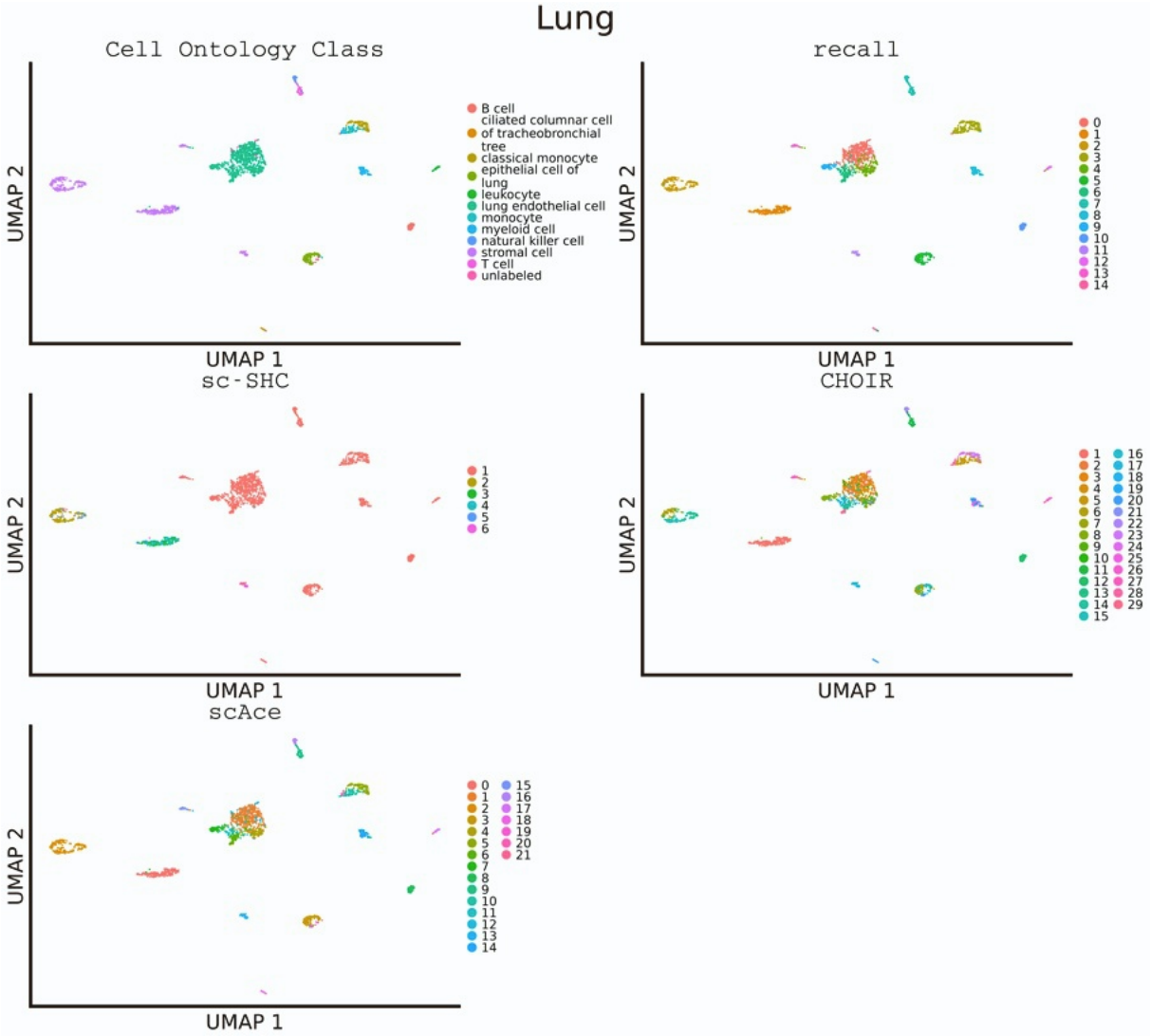


Figure S33. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the lung tissue from the Tabula Muris study.

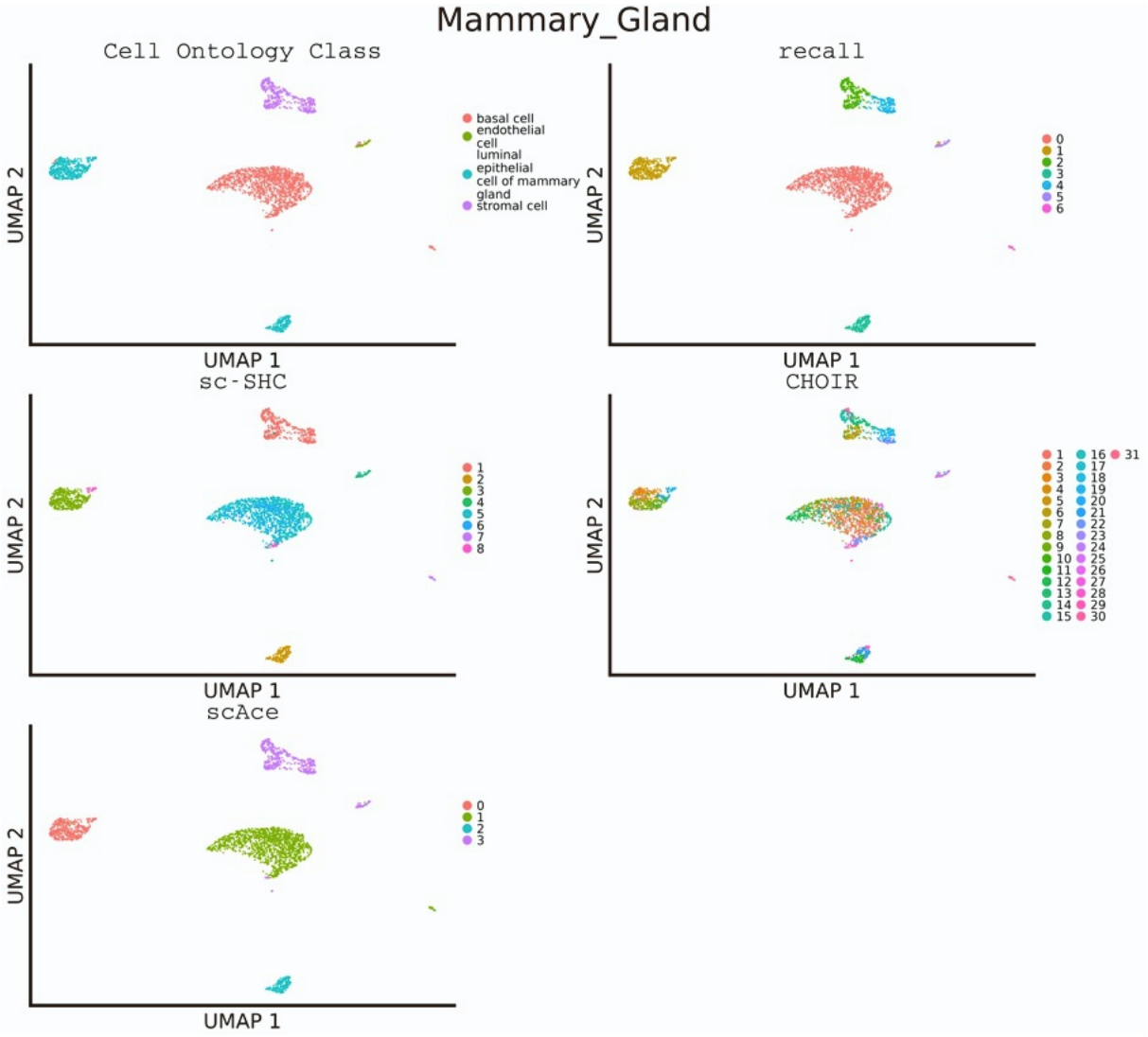


Figure S34. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the mammary gland tissue from the Tabula Muris study.

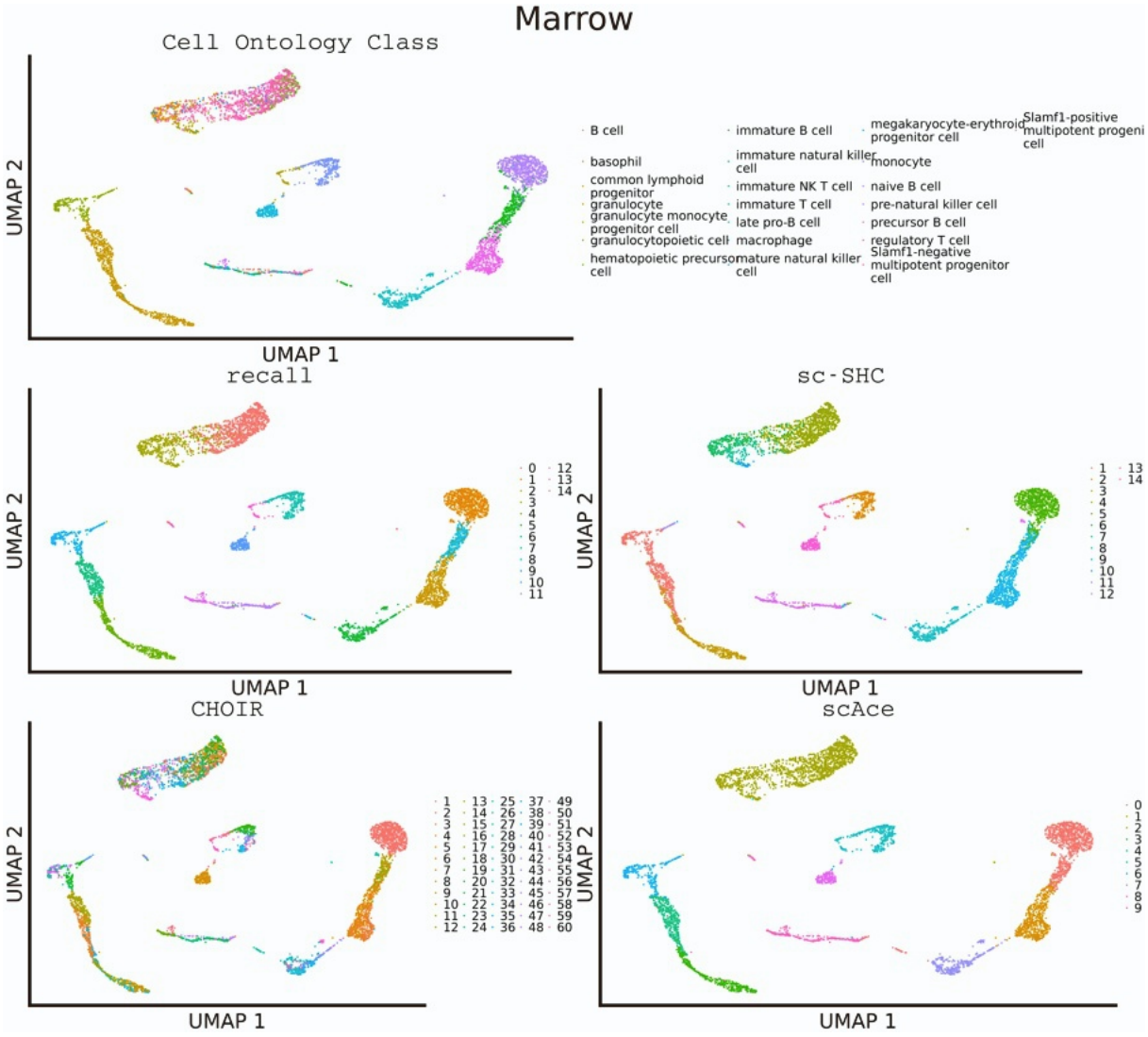


Figure S35. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the bone marrow tissue from the Tabula Muris study.

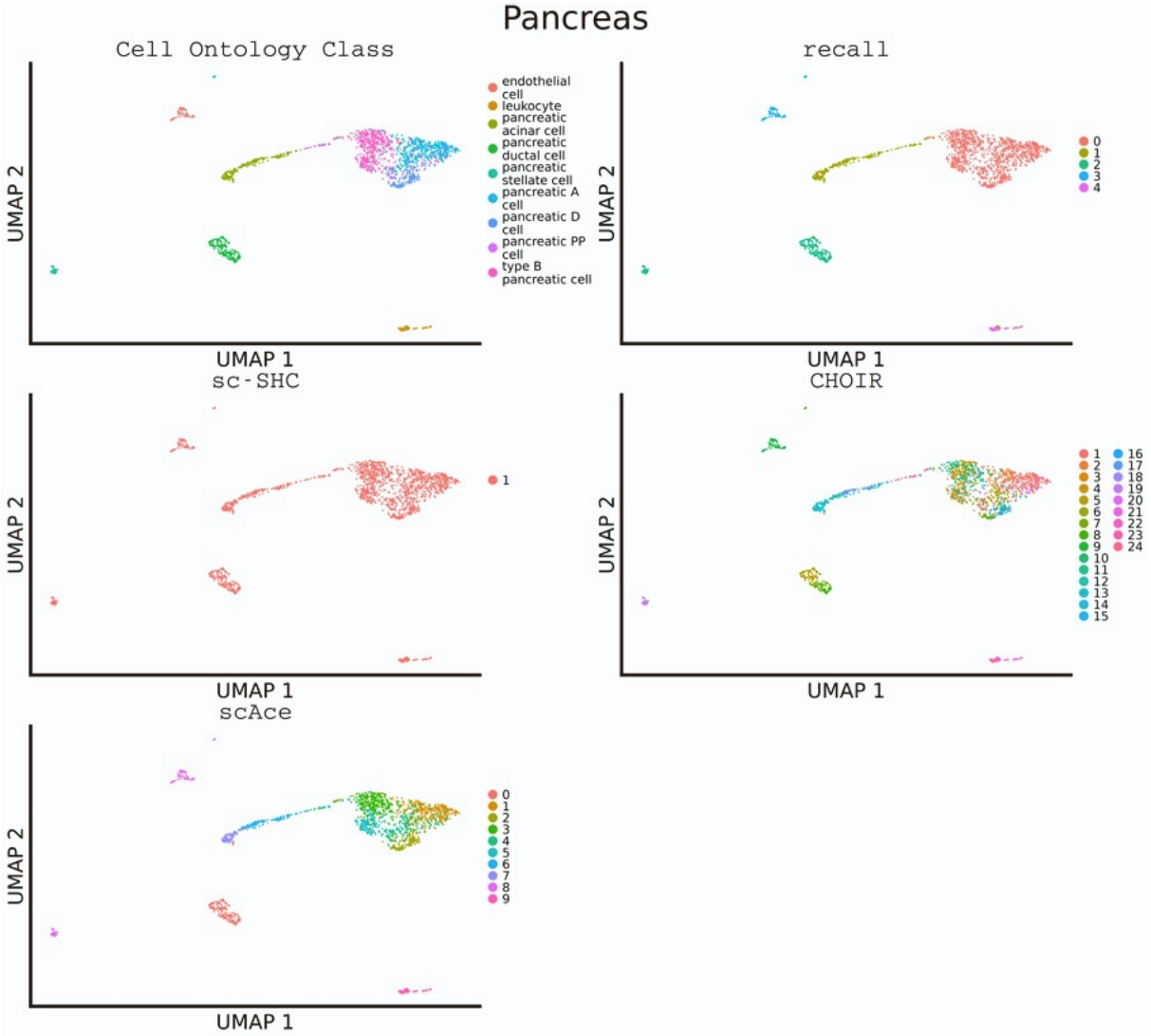


Figure S36. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the pancreas tissue from the Tabula Muris study.

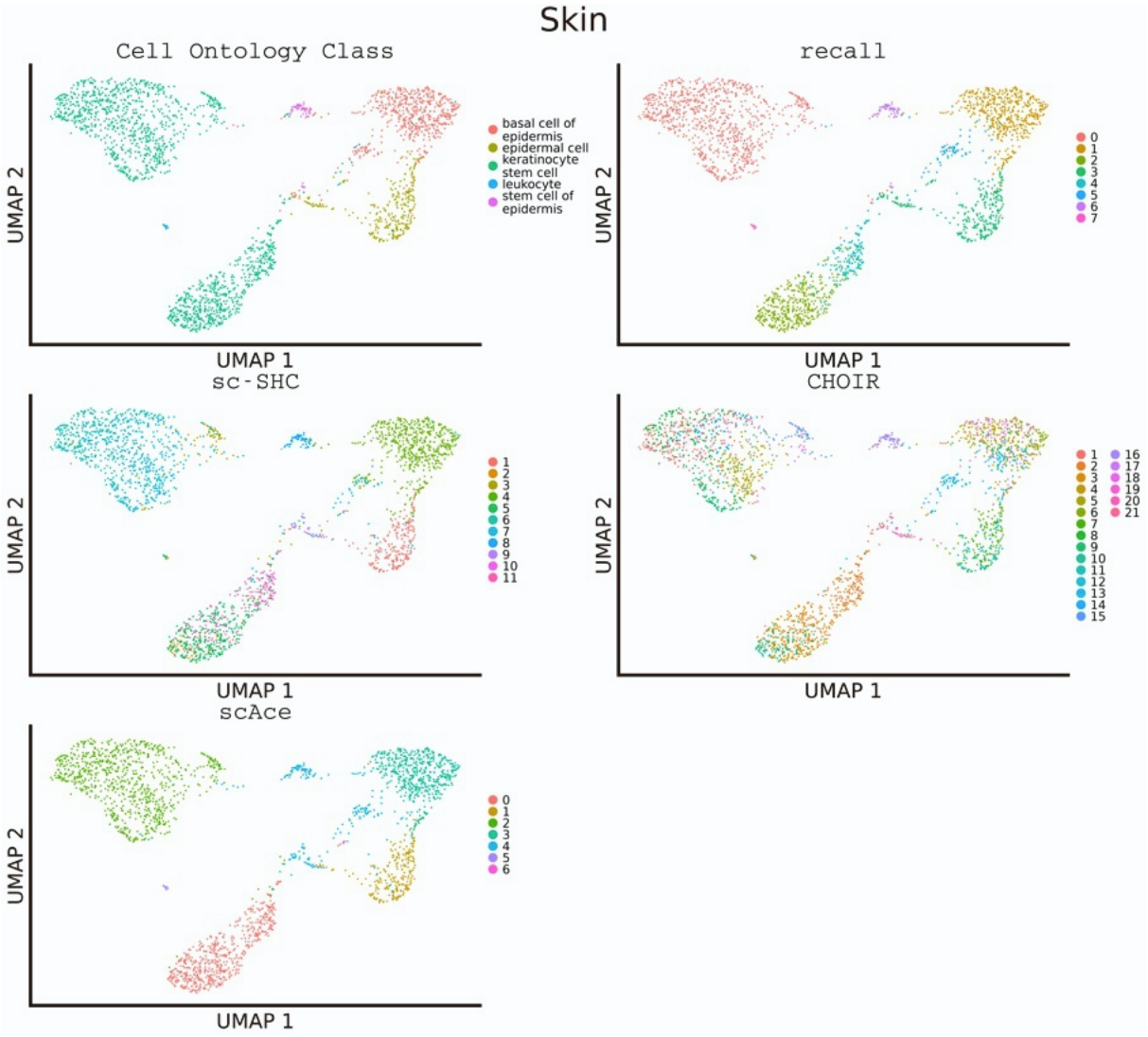


Figure S37. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the skin tissue from the Tabula Muris study.

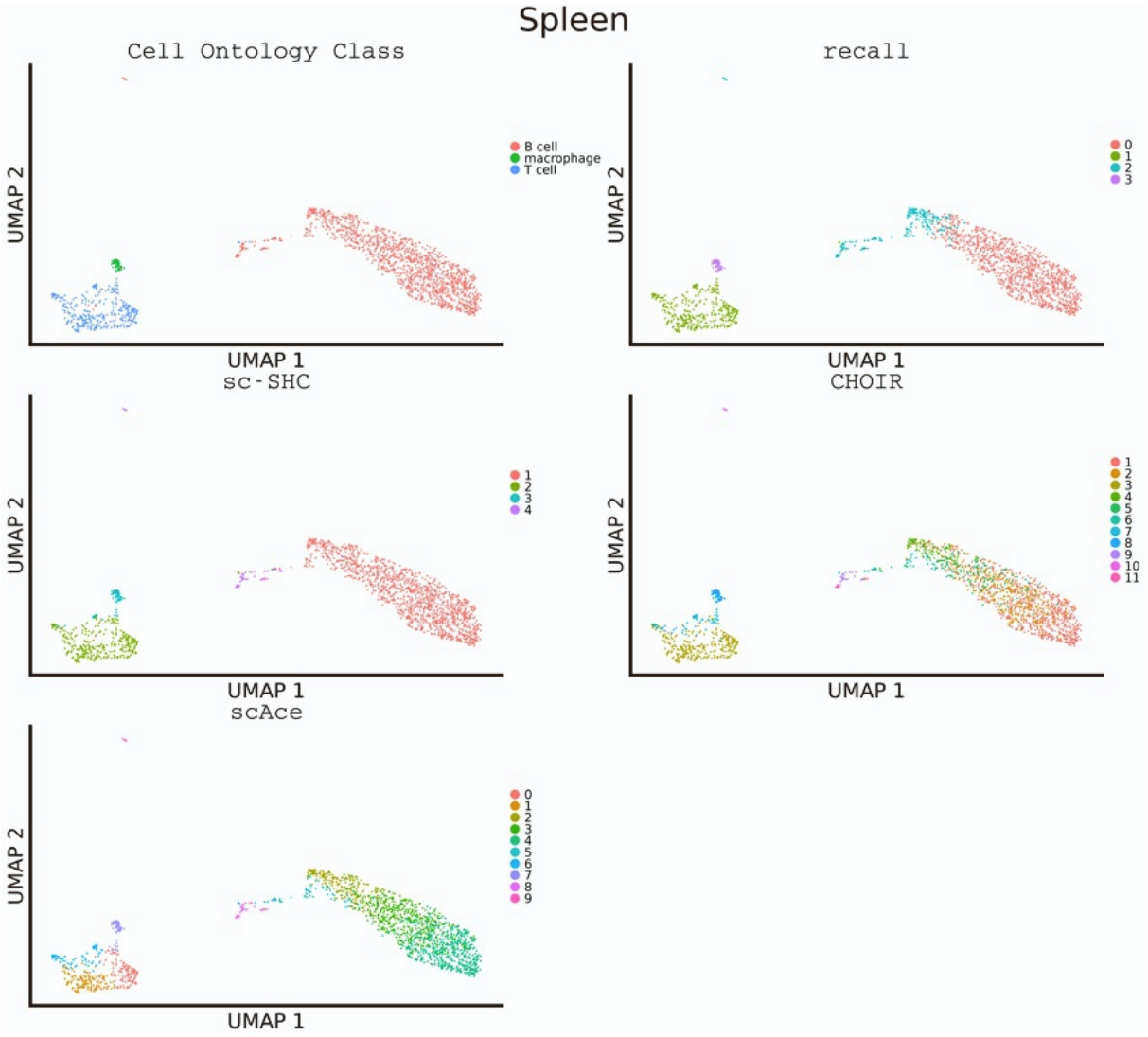


Figure S38. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the spleen tissue from the Tabula Muris study.

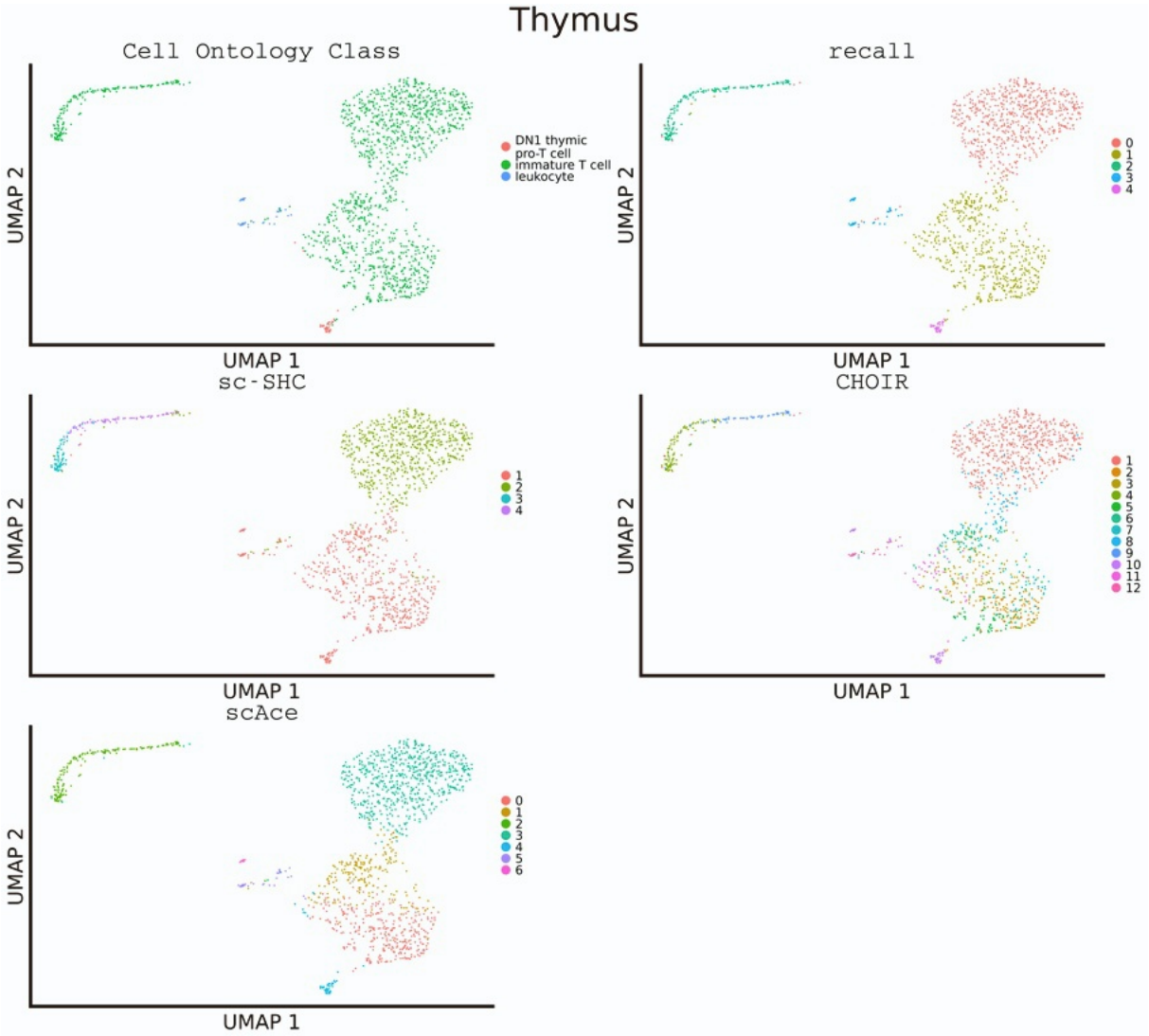


Figure S39. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the thymus tissue from the Tabula Muris study.

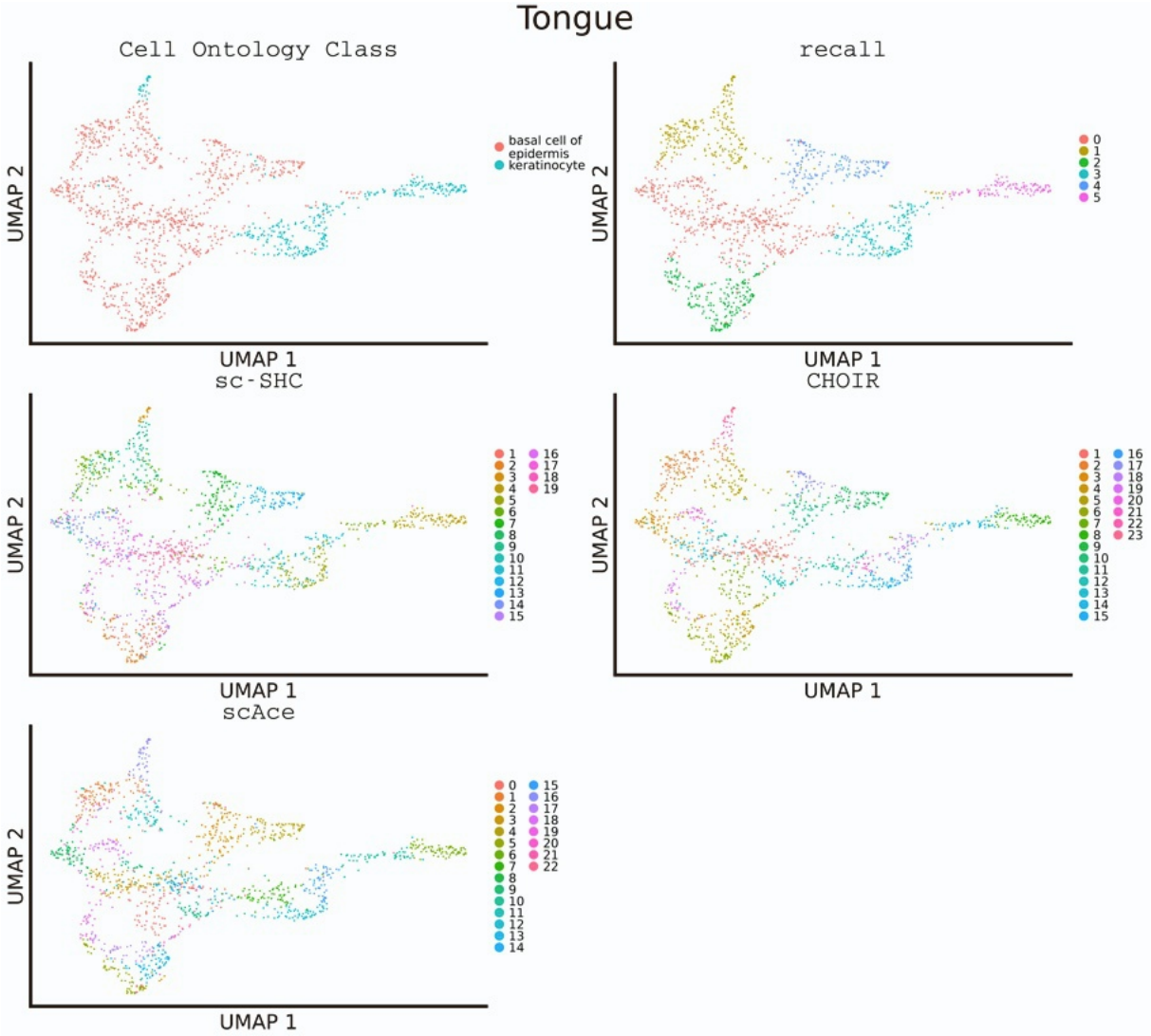


Figure S40. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the tongue tissue from the Tabula Muris study.

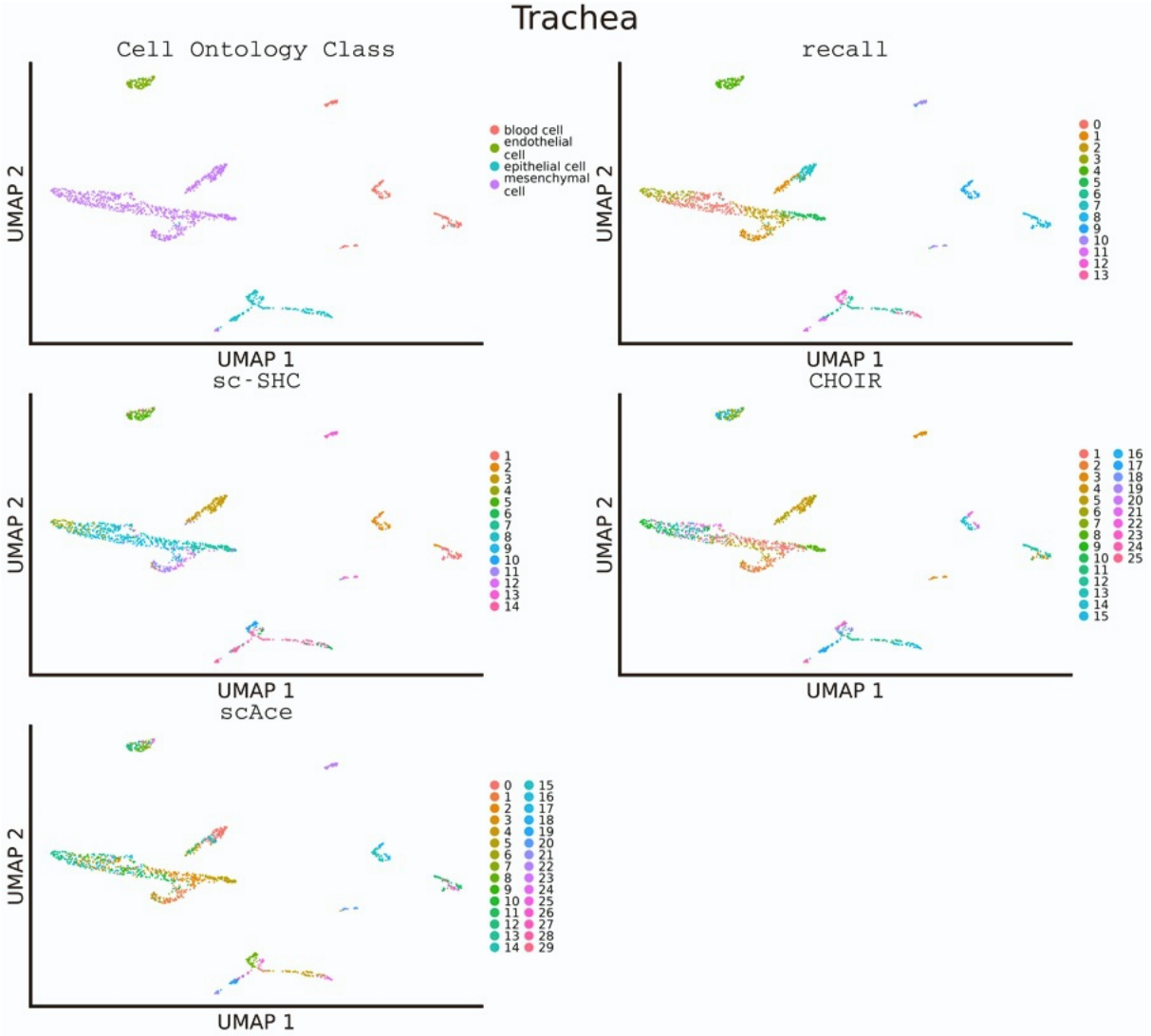


Figure S41. Uniform manifold approximation and projection (UMAP) plots illustrating the manually curated cell ontology class labels compared to the inferred clustering results for `recall+ZIP`, `sc-SHC`, `CHOIR`, and `scAce` when analyzing the trachea tissue from the Tabula Muris study.

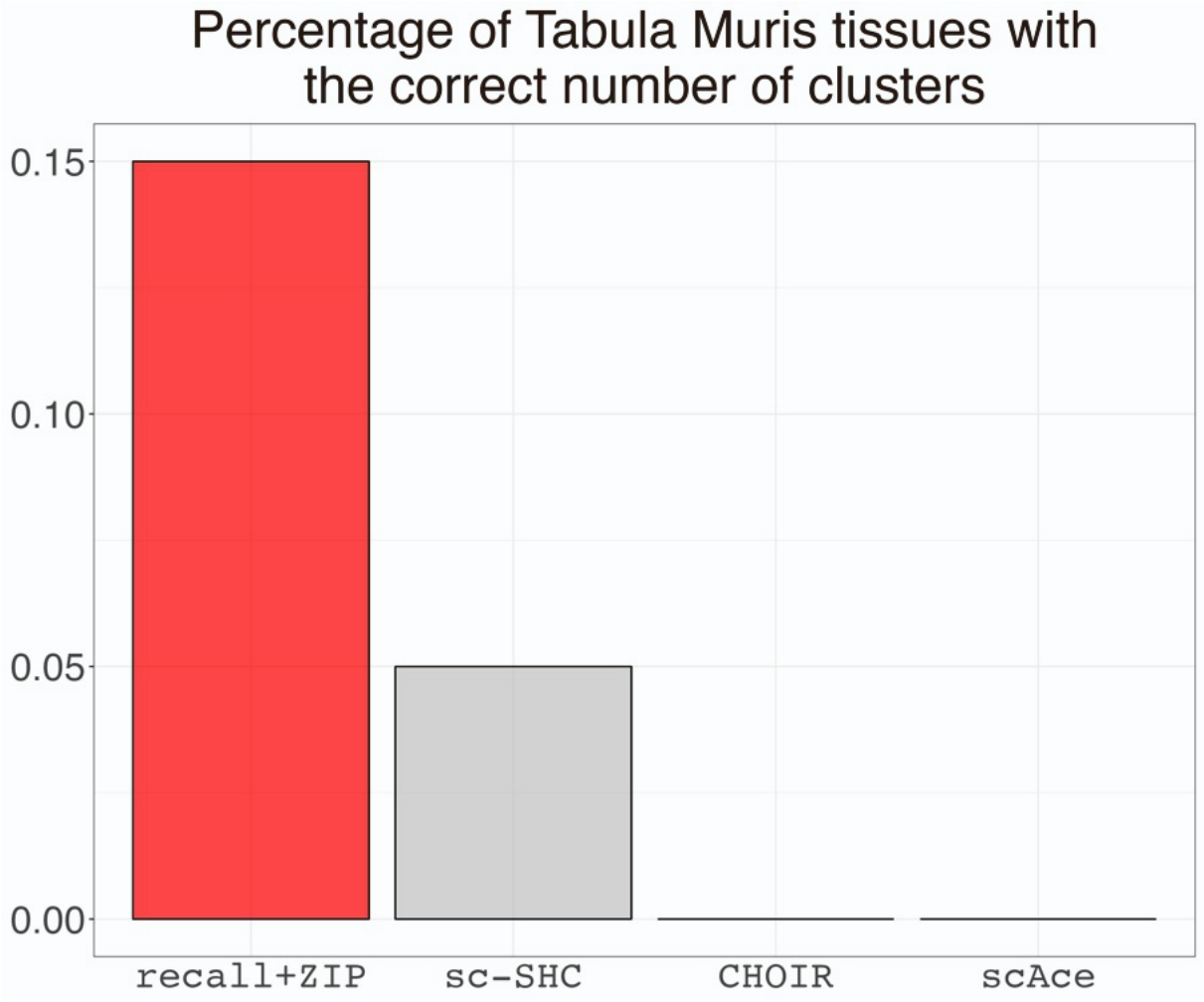


Figure S42. Performance comparison of `recall`, `sc-SHC`,`CHOIR` and `scAce` showing the percentage of tissues in the Tabula Muris dataset where the number of clusters inferred by each method matched the exact number curated cell type groups in the study. Here, `recall` matched the correct number of clusters in 3 tissues (aorta, diaphragm, and limb muscle), `sc-SHC` matched the correct number of clusters in 1 tissue (mammary gland), while `CHOIR` and `scAce` both matched the correct number of clusters in 0 tissues.

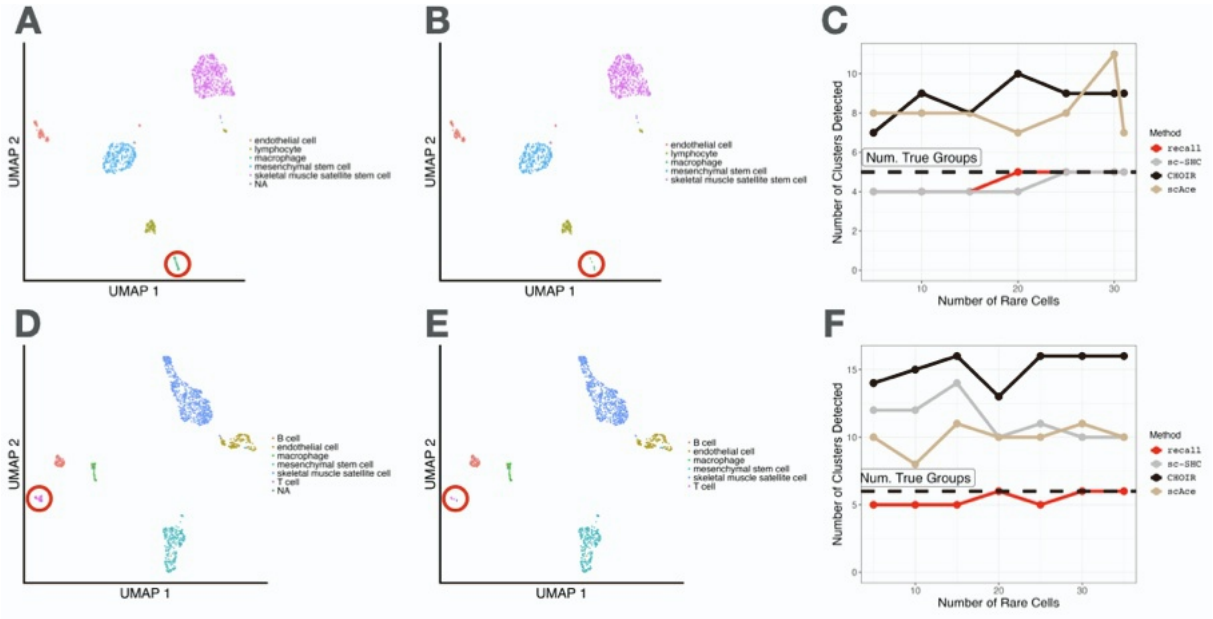


Figure S43. Simulation of rare cell type detection in Tabula Muris tissues. **(a)** Uniform manifold approximation and projection (UMAP) plot showing the limb muscle tissue cells before any downsampling was performed. The circled cell type, macrophages, were selected to be downsampled because they were the cell type with the fewest number of cells (31). **(B)** UMAP plot showing the limb muscle tissue cells after downsampling was performed on the circled cell type, macrophages. This cell type was downsampled to 5, 10, 15, 25, and 30 cells and analyzed by each clustering method. Depicted here is a UMAP with 5 macrophages. **(C)** The number of clusters detected by each method on the downsampled limb muscle datasets. Perfect performance is when a given bar is touching the dashed line. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the macrophages. For `recall` this threshold was 20 cells and for `sc-SHC` this threshold was 25 cells. `CHOIR` and `scAce` sometimes found more clusters after downsampling. **(D)** UMAP plot showing the diaphragm tissue cells before any downsampling was performed. The circled cell type, T cells, were selected to be downsampled because they were the cell type with the fewest number of cells (35). **(E)** UMAP plot showing the diaphragm tissue cells after downsampling was performed on the circled cell type, T cells. This cell type was downsampled to 5, 10, 15, 25, and 30 cells and analyzed by each clustering method. Depicted here is a UMAP with 5 T cells. **(F)** The number of clusters detected by each method on the downsampled diaphragm datasets. Perfect performance is when a given bar is touching the dashed line. The point of this analysis is to identify the threshold at which each method will no longer be able to detect the T cells. For `recall` this threshold was 20 cells (although `recall` also did not detect the T cell cluster with 25 cells). `sc-SHC`, `CHOIR`, and `scAce` sometimes found more clusters after downsampling. and sometimes found two clusters less than they started with.

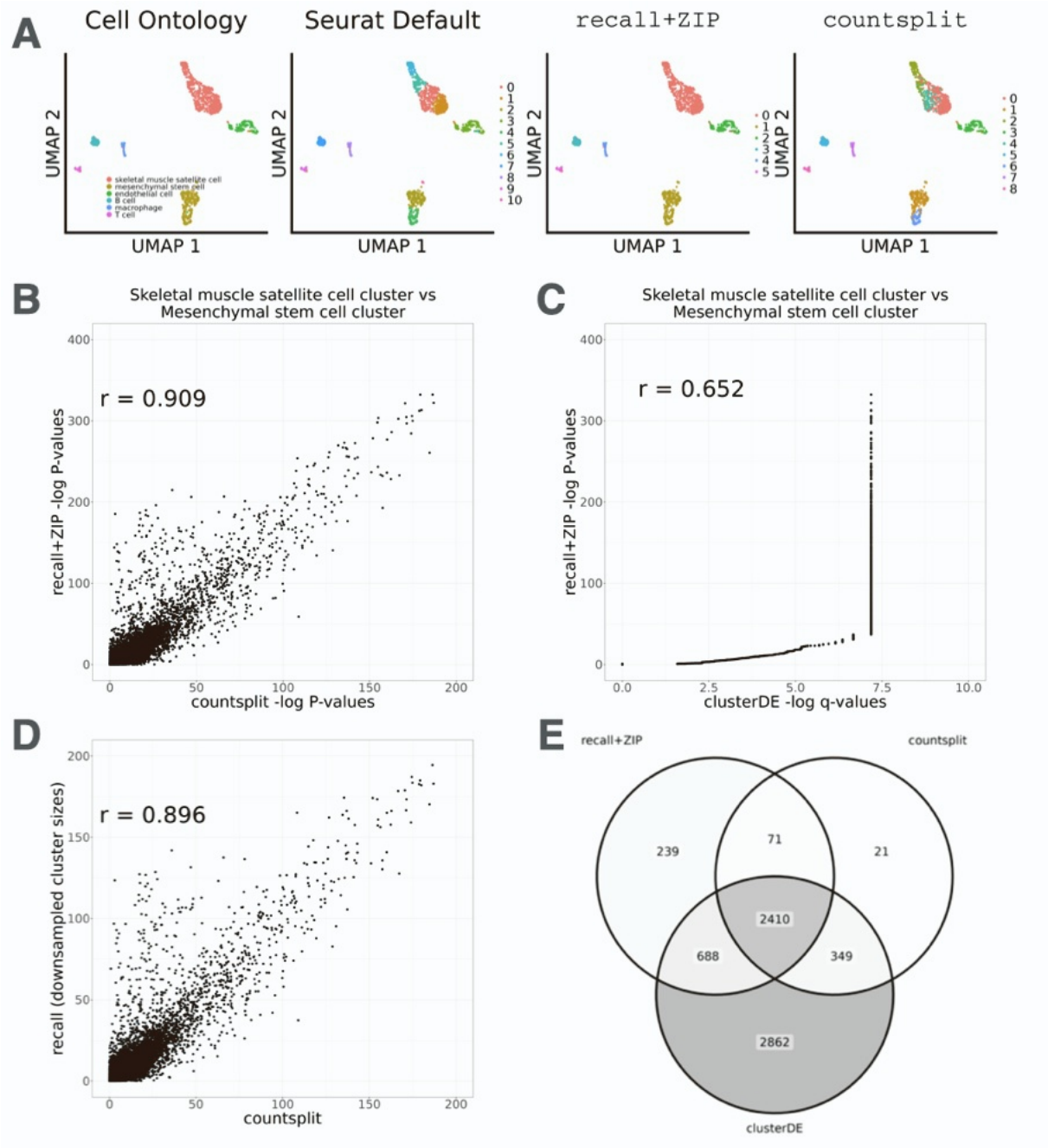


Figure S44. (Continued on the following page).

Figure S44. Differential expression results for `recall+ZIP` compared to `countsplit` and `ClusterDE`, which provide calibrated statistical tests for double-dipping. (A) Uniform manifold approximation and projection (UMAP) plots of the manually curated cell ontology class labels, the default clusters from `Seurat` (used by `ClusterDE`), the cluster labels obtained by `recall+ZIP`, and the cluster labels obtained by `countsplit`. **(B)** Scatter plot and corresponding Pearson correlation coefficient (r) of the -$\log_{10} P$-values for all genes being tested for differential expression between `countsplit` clusters 0 and 1 and for `recall+ZIP` clusters 0 and 1 (both pairs correspond to skeletal muscle satellite cells and mesenchymal stem cells, respectively). **(C)** Scatter plot and corresponding Pearson correlation coefficient (r) of the -$\log_{10} q$-values for all genes being tested for differential expression by `ClusterDE` between default `Seurat` clusters 0 and 1 and the -$\log_{10} P$-values for `recall+ZIP` clusters 0 and 1 (both pairs correspond to skeletal muscle satellite cells and mesenchymal stem cells, respectively). **(D)** Scatter plot and corresponding Pearson correlation coefficient (r) of the -$\log_{10} P$-values for all genes being tested for differential expression between `countsplit` clusters 0 and 1 and for `recall+ZIP` clusters 0 and 1. Here, the `recall+ZIP` clusters had been downsampled such that they were of equal size to the clusters used in `countsplit` (both pairs correspond to skeletal muscle satellite cells and mesenchymal stem cells, respectively). Notice that the $y = x$ line in this comparison better aligns with both algorithms, indicating that any additional significance seen in the -$\log_{10} P$-values from `recall+ZIP` in panel **B** is likely due to the increased sample size of each cluster. This also means that the P-values obtained by testing for differential expression with `recall+ZIP` closely match what would be seen if the cell types were known *a priori*. This is because the `recall+ZIP` clusters align better the curated labels in panel **A**. **(E)** Venn diagram displaying the overlap of statistically significant marker genes found between the comparisons made in **(B)** and **(C)**. `ClusterDE` tests at a false discovery rate (FDR) of 0.05, while `recall+ZIP` and `countsplit` utilize a more strict Bonferroni correction threshold.

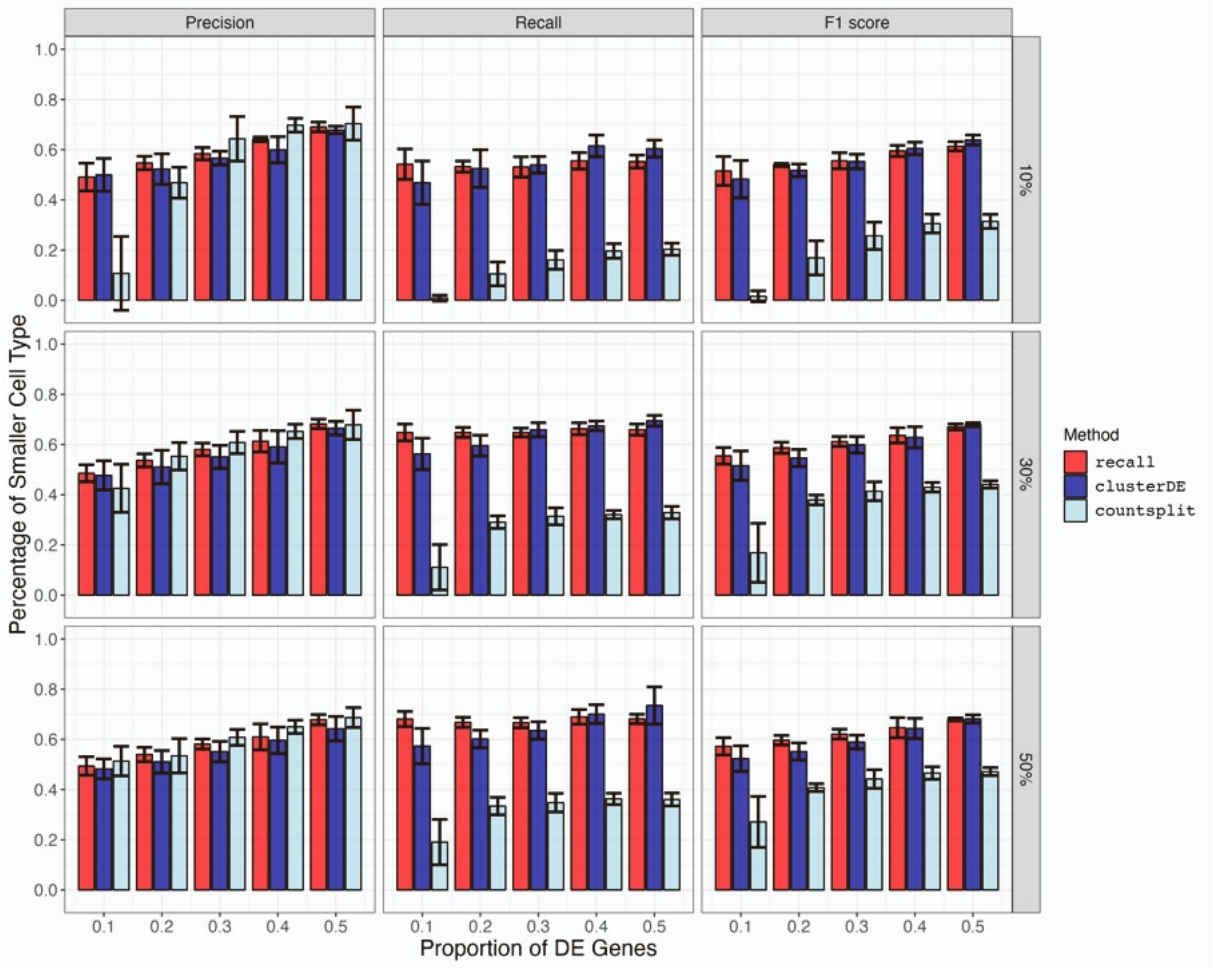


Figure S45. Performance comparison of `recall`, `clusterDE`, and `countsplit` to detect truly differentially expressed genes on simulated datasets with two cell types. Each simulated dataset had five replicates and consisted of $N = 1$K cells and $G = 1$K genes. We considered three different scenarios where we split the cells into 90/10, 70/30, and 50/50 groups of two. The y-axis labels represent the percentage of the less dominant group. Each cell type was simulated such that it had 0.1, 0.2, 0.3, 0.4, and 0.5 proportion of its total genes be differentially expressed. Shown are the precision, recall (sensitivity), and F1 score for each method. Perfect performance is 1.0 for each metric. When the proportion of differentially expressed genes was low, the `FindClusters` function in `Seurat` would often over-cluster when paired with `countsplit`.

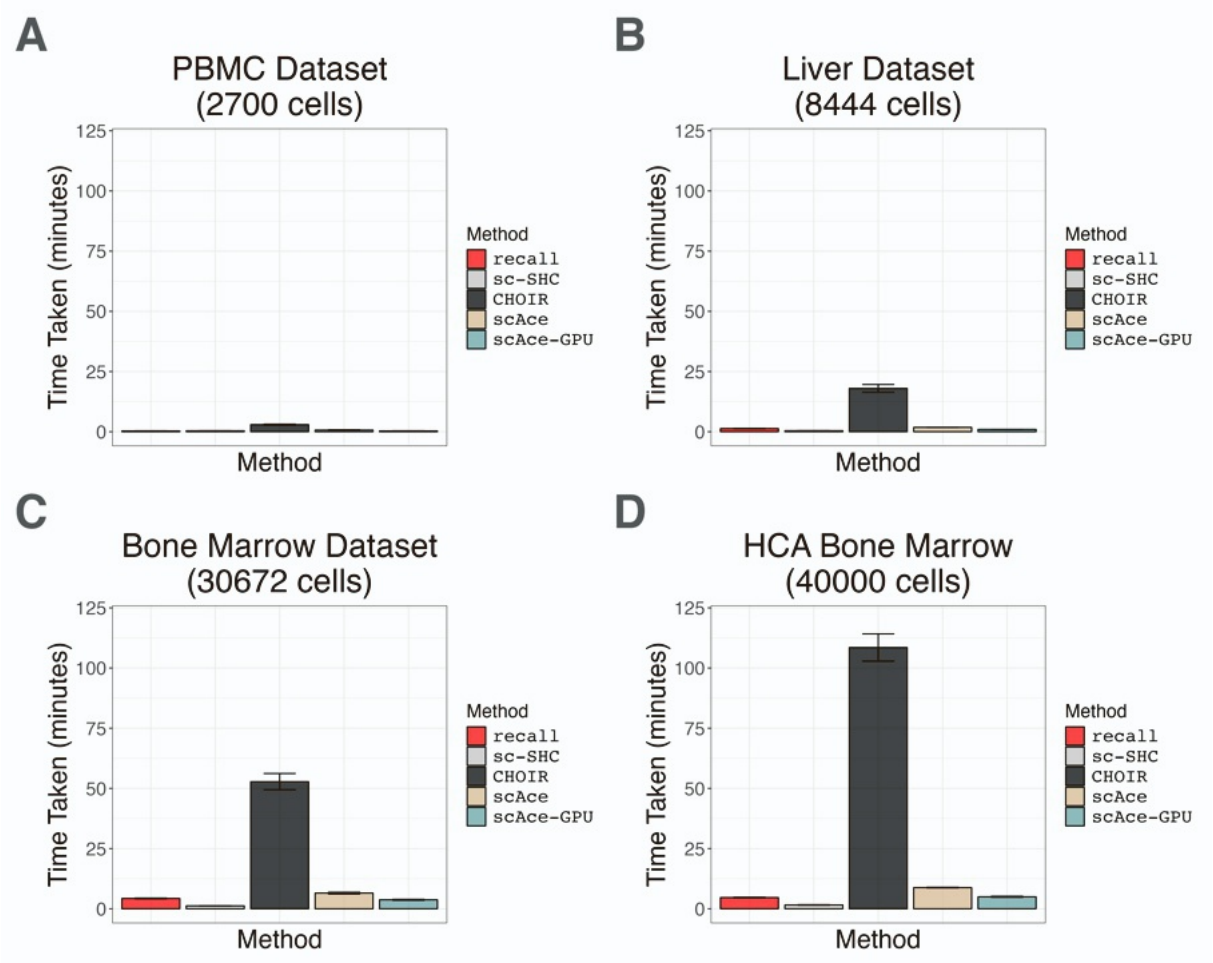


Figure S46. Average runtime comparison of `recall+ZIP`, `sc-SHC`, `CHOIR`, and scAce in four additional single-cell studies of varying sizes. Each method was run on a machine with 16 cores. The datasets analyzed include **(A)** the PBMC 3K (N = 2,700 cells), **(B)** the human liver data from MacParland et al. [9] (N = 8,444 cells), and **(C, D)** the `SeuratData` bone marrow datasets (N = 30,672 and N = 40,000 cells, respectively). We run each method on each dataset 5 times; depicted in each bar plot is the mean ± the standard deviation across all runs.

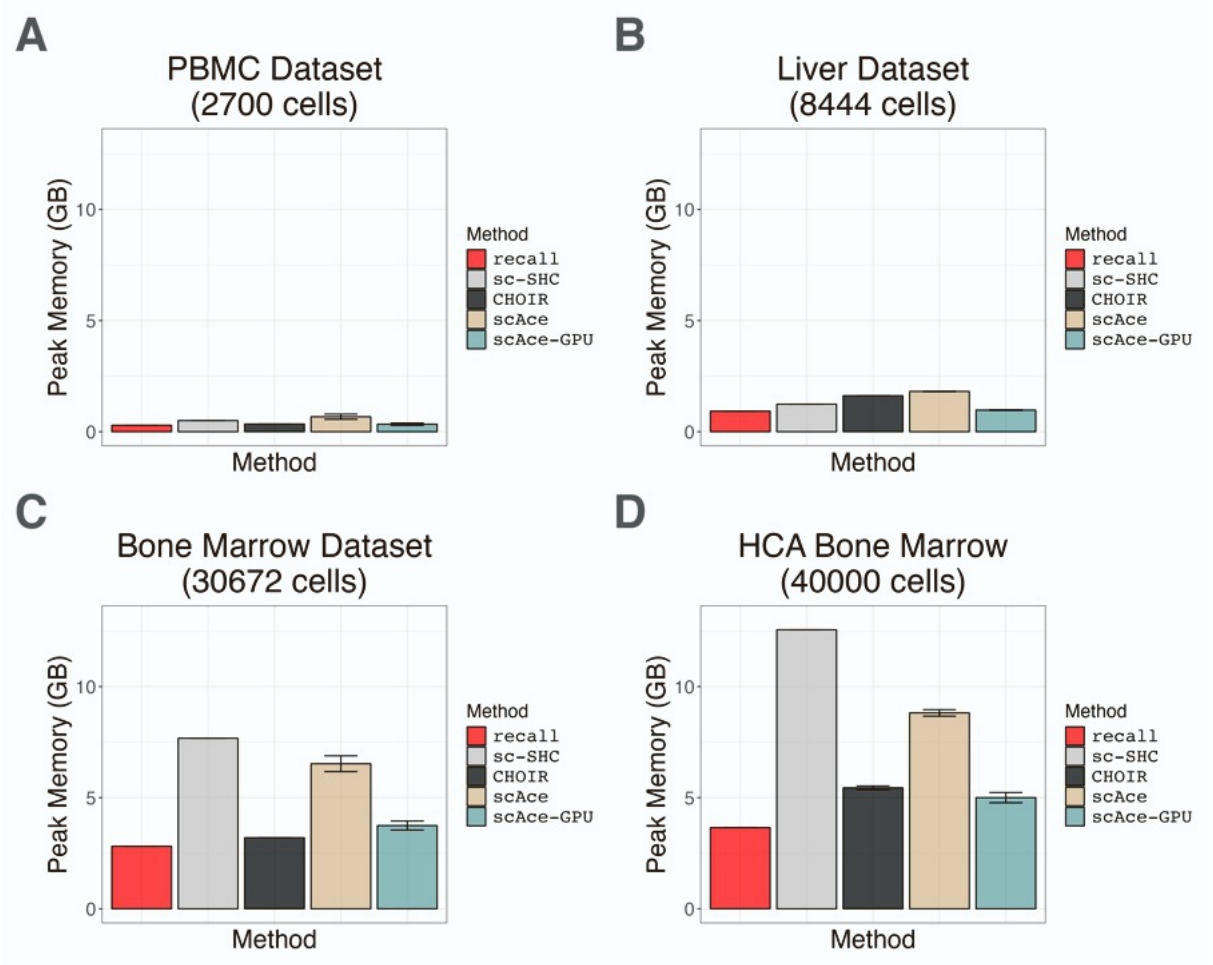


Figure S47. Comparison of peak memory usage (in gigabytes; GB) for `recall+ZIP`, `sc-SHC`, `CHOIR`, and scAce on four additional single-cell studies of varying sizes. Each method was run on a machine with 16 cores. The datasets analyzed include **(A)** the PBMC 3K (N = 2,700 cells), **(B)** the human liver data from MacParland et al. [9] (N = 8,444 cells), and **(C, D)** the `SeuratData` bone marrow datasets (N = 30,672 and N = 40,000 cells, respectively). We run each method on each dataset 5 times; depicted in each bar plot is the mean ± the standard deviation across all runs.

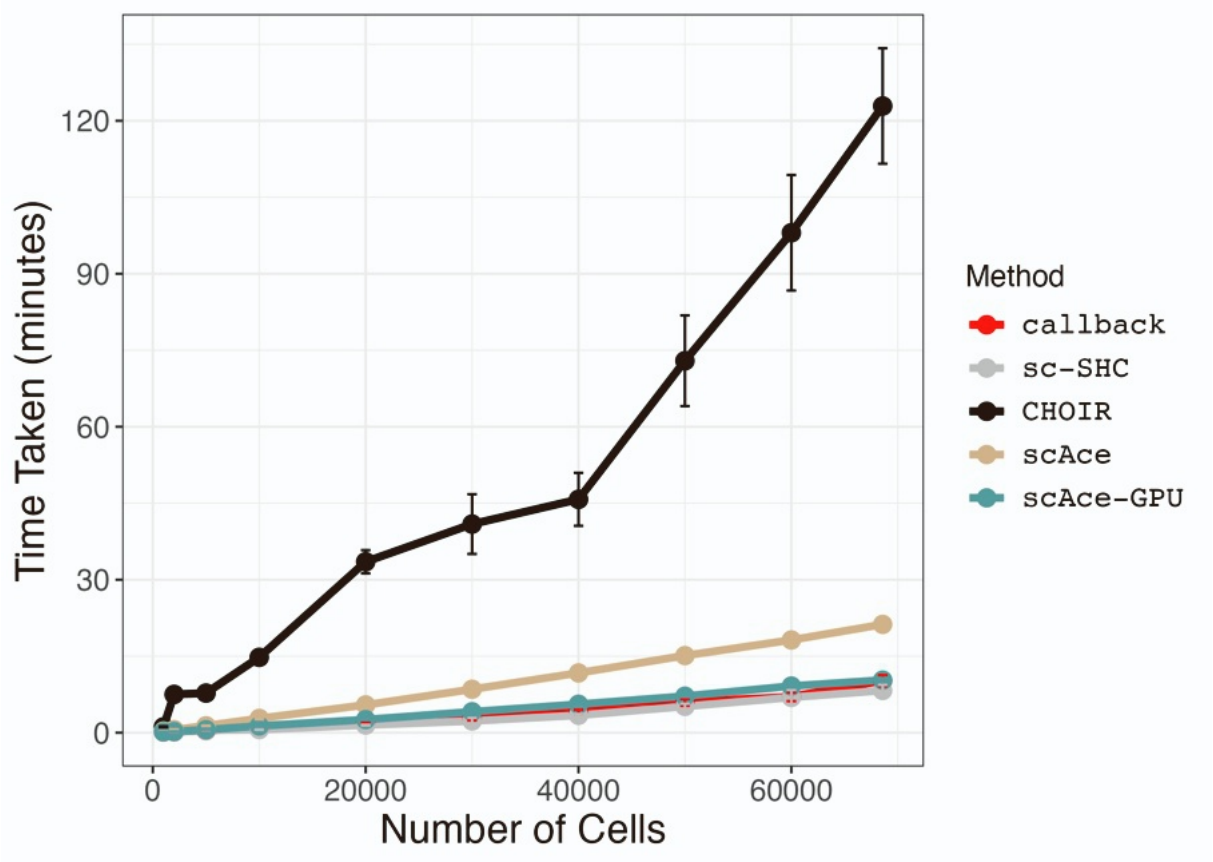


Figure S48. Average runtime comparison of `recall+ZIP`, `sc-SHC`, `CHOIR`, and scAce as a function of the number of cells in a study. Here, we take subsets of the 68,579 total peripheral blood mononuclear cells (PBMCs) provided by Zheng et al. [10] which included smaller datasets of size 1K, 2K, 5K, 10K, 20K, 30K, 40K, 50K, and 60K cells. Each method was run on a machine with 16 cores. We run each method on each dataset 5 times; depicted are the mean ± the standard deviation across all runs.

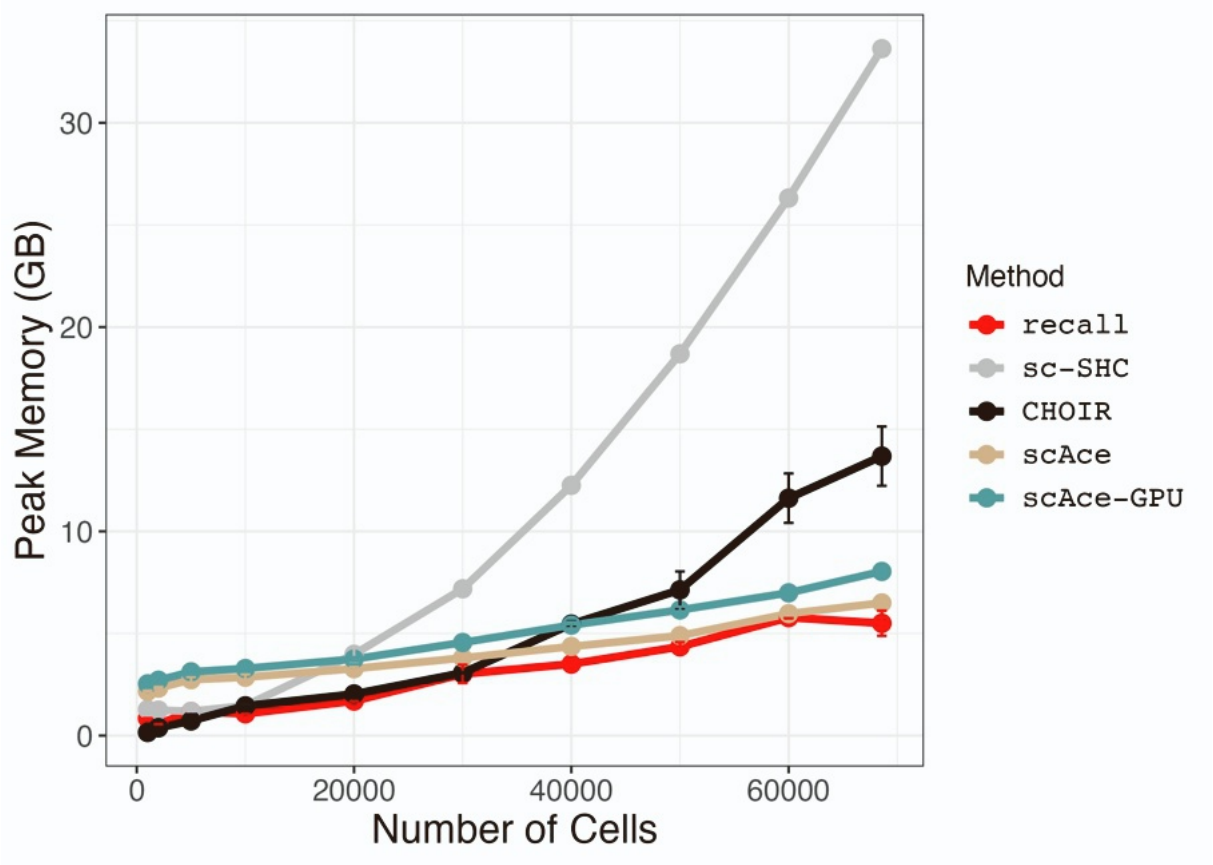


Figure S49. Comparison of peak memory usage (in gigabytes; GB) for `recall+ZIP`, `sc-SHC`, `CHOIR`, and scAce as a function of the number of cells in a study. Here, we take subsets of the 68,579 total peripheral blood mononuclear cells (PBMCs) provided by Zheng et al. [10] which included smaller datasets of size 1K, 2K, 5K, 10K, 20K, 30K, 40K, 50K, and 60K cells. Each method was run on a machine with 16 cores. We run each method on each dataset 5 times; depicted are the mean ± the standard deviation across all runs.

References

1. José E. Chacón and Ana I. Rastrojo. Minimum adjusted rand index for two clusterings of a given size. *Advances in Data Analysis and Classification*, 17(1):125–133, Mar 2023. ISSN 1862-5355. doi: 10.1007/s11634-022-00491-w. URL https://doi.org/10.1007/s11634-022-00491-w.

2. William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. ISSN 01621459. URL http://www.jstor.org/stable/2284239.

3. E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983. doi: 10.1080/01621459.1983.10478008. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1983.10478008.

4. Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In Jason Eisner, editor, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1043.

5. Lijia Yu, Yue Cao, Jean Y. H. Yang, and Pengyi Yang. Benchmarking clustering algorithms on estimating the number of cell types from single-cell rna-sequencing data. *Genome Biology*, 23(1):49, Feb 2022. ISSN 1474-760X. doi: 10.1186/s13059-022-02622-0. URL https://doi.org/10.1186/s13059-022-02622-0.

6. Norman Lloyd. Johnson, Samuel. Kotz, and Adrienne W. Kemp. *Univariate discrete distributions*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. John Wiley & Sons, New York, 2nd ed. edition, 1992. ISBN 0471548979. Includes bibliographical references (p. 473-547) and index.

7. Stefanie Dencks, Marion Piepenbrock, and Georg Schmitz. Assessing vessel reconstruction in ultrasound localization microscopy by maximum likelihood estimation of a zero-inflated poisson model. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 67(8):1603–1612, 2020. doi: 10.1109/TUFFC.2020.2980063.

8. Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, William M. Mauck III, Shiwei Zheng, Andrew Butler, Maddie J. Lee, Aaron J. Wilk, Charlotte Darby, Michael Zagar, Paul Hoffman, Marlon Stoeckius, Efthymia Papalexi, Eleni P. Mimitou, Jaison Jain, Avi Srivastava, Tim Stuart, Lamar B. Fleming, Bertrand Yeung, Angela J. Rogers, Juliana M. McElrath, Catherine A. Blish, Raphael Gottardo, Peter Smibert, and Rahul Satija. Integrated analysis of multimodal single-cell data. *Cell*, 2021. doi: 10.1016/j.cell.2021.04.048. URL https://doi.org/10.1016/j.cell.2021.04.048.

9. Sonya A. MacParland, Jeff C. Liu, Xue-Zhong Ma, Brendan T. Innes, Agata M. Bartczak, Blair K. Gage, Justin Manuel, Nicholas Khuu, Juan Echeverri, Ivan Linares, Rahul Gupta, Michael L. Cheng, Lewis Y. Liu, Damra Camat, Sai W. Chung, Rebecca K. Seliga, Zigong Shao, Elizabeth Lee, Shinichiro Ogawa, Mina Ogawa, Michael D. Wilson, Jason E. Fish, Markus Selzner, Anand Ghanekar, David Grant, Paul Greig, Gonzalo Sapisochin, Nazia Selzner, Neil Winegarden, Oyedele Adeyi, Gordon Keller, Gary D. Bader, and Ian D. McGilvray. Single cell rna sequencing of human liver reveals distinct intrahepatic macrophage populations. *Nature Communications*, 9(1):4383, Oct 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-06318-7. URL https://doi.org/10.1038/s41467-018-06318-7.

10. Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8(1):14049, Jan 2017. ISSN 2041-1723. doi: 10.1038/ncomms14049. URL `https://doi.org/10.1038/ncomms14049`.