Supplementary Information for the article: “EnImpute: imputing dropout events in single cell RNA sequencing data via ensemble learning"

Xiao-Fei Zhang, Le Ou-Yang, Shuo Yang, Xing-Ming Zhao, Xiaohua Hu and Hong Yan

1 Details of EnImpute

1.1 Overview of EnImpute

EnImpute combines the results obtained from multiple individual imputation methods. The current implementation of EnImpute integrates eight state-of-the-art methods: Adaptively-thresholded low rank approximation (ALRA) [1], Deep count autoencoder network (DCA) [2], DrImpute [3], Markov affinity-based graph imputation of cells (MAGIC) [4], Single-cell analysis via expression recovery (SAVER) [5], scImpute [6], scRMD [7], and Seurat [8]. The overview of EnImpute is presented in Figure 1 in the main text. We first briefly review the eight imputation methods, and then present how to consolidate the results generated by different base methods into a consensus result.

1.2 Individual imputation methods

The eight imputation methods integrated in EnImpute are summarized in Table S1. Main features of each method are highlighted as follows.

Table S1: Summary of imputation methods integrated in EnImpute

Methods	Keyword descriptions	Website	Language
ALRA [1]	Low rank approximation	https://github.com/KlugerLab/ALRA/	R
DCA [2]	Zero-inflated negative binomial, autoencoder	https://github.com/theislab/dca	Python
DrImpute [3]	PCA, ensemble clustering, average	https://github.com/gongx030/DrImpute	R
MAGIC [4]	PCA, Markov affinity, data diffusion	https://github.com/KrishnaswamyLab/magic	R, Python
SAVER [5]	Poisson-gamma mixture, Poisson LASSO regression	https://github.com/mohuangx/SAVER	R
scImpute [6]	PCA, spectral clustering, mixture model, least squares	https://github.com/Vivianstats/scImpute	R
scRMD [7]	robust matrix decomposition, low rank, sparse	https://github.com/ChongC1990/scRMD	R
Seurat [8]	L1-constrained linear models	https://github.com/satijalab/seurat	R

1.2.1 ALRA

Adaptively-thresholded Low Rank Approximation (ALRA) [1] is a method for imputing dropout values in scRNA-seq data. It uses low-rank approximation to replace false zero values caused by dropout events, and uses an adaptively-threshold method to preserve biologically non-expressed genes at zeros. It includes the following steps: (i) It performs library normalization on a given expression matrix and takes a log transformation to obtain the normalized expression matrix; (ii) The low rank approximation of the normalized expression matrix is computed using the R package rsvd; (iii) Each gene of the low rank approximation matrix is thresholded by the absolute value of that gene's most negative entry; (iv) The resulting matrix is rescaled so that the mean and standard deviation of each gene in the resulting matrix match that of the original expression matrix. The key tuning parameter of ALRA is the rank k used in the low rank approximation. ALRA develops a heuristic method to choose k based on the statistics of the spacings between consecutive singular values.

1.2.2 DCA

Deep count autoencoder network (DCA) [2] is an autoencoder-based method to impute the false zero expressions in scRNA-seq data. Unlike traditional autoencoders, DCA uses a zero-inflated negative binomial distribution to model the count distribution, overdispersion and sparsity of scRNA-seq data. To avoid overfitting and overimputation, several regularization methods, including dropout, encoder-specific and overall L1 and L2 regularization are implemented. DCA also implements a parameter search approach (the number of layers, the number of neurons, and the L1/L2 coefficients). In addition, the runtime of DCA is linear with the number of cells, and it has a considerable speed advantage over most imputation methods. The key parameter of DCA is the type of autoencoder (loss function).

1.2.3 DrImpute

DrImpute [3] is an ensemble clustering based imputation method. It includes the following steps: (i) It first normalizes the raw read counts by library size factor and then takes a log transformation; (ii) Two similarity matrices among cells are computed using the Pearson and Spearman correlations; (ii) K-means clustering with different values of K (the predefined number of clusters) are implemented on the first 5% of the principal components of the similarity matrices; (iv) For each base clustering result, the imputation values are calculated by averaging each cluster; (v) The final imputation values are obtained by averaging the results from different base clusterings. The key tuning parameter of DrImpute is the number of cell groups when the K-means clustering is run.

1.2.4 MAGIC

Markov affinity-based graph imputation of cells (MAGIC) [4] imputes dropout events by borrowing information from similar cells via data fusion. MAGIC includes the following steps: (i) It normalizes the library size for the count matrix and runs PCA on the normalized data; (ii) The affinity matrix from PCA-transformed data is constructed using an adaptive Gaussian Kernel and rendered into a Markov transition matrix via row-stochastic normalization; (iii) The t-step transition matrix is computed through exponentiation of the Markov transition matrix; (iv) The data is imputed via matrix product between the t-step transition matrix and the library-size normalized count matrix. Diffusion time for Markov affinity matrix t is a key parameter in MAGIC. MAGIC develops a method to select t adaptively based on the degree of change between the imputed data at time t and time $t-1$.

1.2.5 SAVER

Single-cell analysis via expression recovery (SAVER) [5] uses gene-to-gene relationships to impute dropout events in scRNA-seq data via a Bayesian approach. The input to SAVER is the raw read count matrix. It assumes that the expression level of each gene in each cell follows a Poisson-gamma mixture distribution. It uses a penalized Poisson LASSO regression to estimate the prior mean of gamma distribution in an empirical Bayes-like approach. The prior variance of gamma distribution is estimated by assuming a constant noise model across cells. The posterior gamma distribution of the true expression is computed based on the estimated prior mean and variance parameters. The posterior mean is used as the imputed result.

1.2.6 scImpute

scImpute [6] is a method that simultaneously determines which values are affected by dropout events and performs imputation only on dropout entries. It first learns each gene's dropout probability in each cell using a mixture model, and then imputes the highly probable dropout values. scImpute includes the following steps: (i) It normalizes the library size for the count matrix so that all cells have one million reads, and takes the log transformation; (ii) Cell subpopulations are identified by performing spectral clustering on the PCA-transformed data; (iii) Whether a zero value comes from a dropout event or not is determined using a mixture model; (iv) The dropout zero expressions are imputed using a non-negative least squares regression. scImpute includes two key parameters: K specifying the number of initial clusters and t determining the dropout threshold.

1.2.7 scRMD

scRMD [7] imputes dropout values using a robust matrix decomposition method. It assumes that the underlying true expression matrix is low rank, and that the dropout events are sparse in the expression matrix. Then it decomposes the observed count matrix into a low rank term that captures the true expression values and a sparse term that captures the dropouts. An alternating direction method of multiplier algorithm is proposed to solve the model. The imputed matrix is constructed based on the estimated low rank term and sparse term. scRMD includes two tuning parameters: λ controlling the rank of the low rank term, and τ controlling the sparsity level of the sparse term. A random matrix-based method is employed to choose the two tuning parameters.

1.2.8 Seurat

Seurat [8] imputes the false zero values in scRNA-seq data by borrowing information from similar genes. For each target gene, Seurat constructs a linear model with highly variable genes as predictors. The L1-constrained lasso algorithm is used to solve the model. Then the predicted values are used as the imputed expression values.

1.3 Ensemble combination rule

After obtaining the multiple imputed results from different individual methods, we perform an ensemble method to provide a consensus result. Let $X^{(k)} = \left(x_{ij}^{(k)}\right)$ be a $p \times n$ imputed count matrix generated by the k-th method, where p is the number of genes and n is the number of cells. For ALRA, DrImpute, MAGIC, scRMD, and Seurat which generate the imputed count matrices in log-scale, the expression levels are exponentiated. The exponentiated count matrix is defined as $\bar{x}_{ij}^{(k)} = \exp\left(x_{ij}^{(k)}\right) - 1$. For DCA, SAVER and scImpute which generate the imputed count matrices in exponential scale, the exponentiated count matrices are defined as $\bar{X}^{(k)} = X^{(k)}$. Different methods normalize the raw read counts with different size factors in the data preprocessing step. The imputed count matrices produced by different methods are different in scale. Therefore, we rescale the imputed results from different methods by performing library normalization such that library size for each cell is 10,000. That is, the imputed matrices are rescaled as follows $\tilde{x}_{ij}^{(k)} = \frac{\bar{x}_{ij}^{(k)}}{\sum_{i=1}^{p} \bar{x}_{ij}^{(k)}} * 10000$. This effectively eliminates the method-to-method variations in scale. Then the rescaled data are log transformed with pseudocount 1, $\dot{x}_{ij}^{(k)} = \log\left(\tilde{x}_{ij}^{(k)} + 1\right)$. Finally, we use the trimmed mean of imputed values generated by different methods

as the final imputation, $\hat{x}_{ij} = trimmed\_mean\left(\dot{x}_{ij}^{(1)}, \ldots, \dot{x}_{ij}^{(8)}, trim\right)$, where $trim$ (between 0 and 0.5) is the parameter specifying the fraction of observations to be trimmed from each end before the mean is computed. Instead of the standard mean or median which is widely used in ensemble learning, here we use the trimmed mean since it is less sensitive to outliers than the standard mean and uses more information from the observations than the median. Note that if $trim = 0$, the trimmed mean is equal to the standard mean, and if $trim = 0.5$, the trimmed mean is equal to the median.

1.4 Summary of EnImpute

EnImpute is summarized in Algorithm 1

Algorithm 1 Summary of EnImpute

1. Run the eight individual imputation methods to get the base imputed count matrices, $X^{(k)}$ for $k = 1, \ldots, 8$.
2. For ALRA, DrImpute, MAGIC, scRMD, and Seurat, transform the expression levels into exponential scale, $\bar{x}_{ij}^{(k)} = \exp\left(x_{ij}^{(k)}\right) - 1$.
3. Rescale the imputed count matrices to eliminate method-to-method variations in scale, $\tilde{x}_{ij}^{(k)} = \frac{\bar{x}_{ij}^{(k)}}{\sum_{i=1}^{p} \bar{x}_{ij}^{(k)}} * 10000$.
4. Take a log transformation with pseudocount 1, $\dot{x}_{ij}^{(k)} = \log\left(\tilde{x}_{ij}^{(k)} + 1\right)$.
5. Compute the consensus imputed result using the trimmed mean, $\hat{x}_{ij} = trimmed\_mean\left(\dot{x}_{ij}^{(1)}, \ldots, \dot{x}_{ij}^{(8)}, trim\right)$.

2 Implementation, dependencies, installation and usages

We develop an R package, named EnImpute, to implement the ensemble method for imputing dropout values in scRNA-seq data. The base results are generated by ALRA, DCA, DrImpute, MAGIC, SAVER, scImpute, scRMD and Seurat. The R functions available at https://github.com/KlugerLab/ALRA are used to implement ALRA. We first use the function `normalize_data` to normalize (library and log normalization) the raw count matrix, and then use the function `alra` to impute the dropout values. DCA is developed by the authors using Python (https://github.com/theislab/dca). We use the `system` function in R to run DCA from the command line. The R package `DrImpute` available at https://github.com/ikwak2/DrImpute is used to implement DrImpute. The raw data is preprocessed using the function `preprocessSC`, followed by log transformation with pseudocount 1. The function `DrImpute` is used to impute the data. The Python and R packages available at https://github.com/KrishnaswamyLab/MAGIC are used to implement MAGIC. The function `library.size.normalize` in the R packages Rmagic is used to perform normalization on the raw read count. A log transformation with pseudocount 1 is followed. The function `magic` is used to impute the data. The `saver` function in the R package `SAVER` is used to implement SAVER (https://github.com/mohuangx/SAVER).

scImpute is implemented using the scimpute function in the R package scImpute (https://github.com/Vivianstats/scImpute). We use the rmd function in the R package scRMD to implement scRMD (https://github.com/ChongC1990/scRMD). Before imputing, library normalization and log transformation with pseudocount 1 are taken. The AddImputedScore function in the R package Seurat is used to implement Seurat (https://github.com/satijalab/seurat).

After obtaining the base results generated by individual imputation methods, the results generated by ALRA, DrImpute, MAGIC, scRMD and Seurat are exponentiated using the function exp. The imputed results are then normalized by the size factor, followed with log transformations. Finally, the consensus imputed result is obtained by computing the trimmed mean (using the function mean) of the results from different base methods.

The EnImpute package has the following R-package dependencies:

- DrImpute – for implementing DrImpute,
- Rmagic – for implementing MAGIC,
- SAVER – for implementing SAVER,
- scImpute – for implementing scImpute,
- scRMD – for implementing scRMD,
- Seurat – for implementing Seurat,
- rsvd – dependency of ALRA.

These R packages will be automatically installed along with EnImpute.

EnImpute also depends on the following Python packages:

- dca – for implementing DCA,
- MAGIC – for implementing MAGIC.

Before installing the EnImpute package, please install the two Python packages following the corresponding readme files, and check whether they can be run from the command line.

EnImpute is freely available at https://github.com/Zhangxf-ccnu/EnImpute. One can run the following commands in R to install EnImpute from GitHub.

```
# Step 1. Install the devtools package. Invoke R and then type
    install.packages("devtools")
# Step 2. Load the devtools package.
    library("devtools")
# Step 3. Install the EnImpute package from GitHub.
    install_github("Zhangxf-ccnu/EnImpute", subdir="pkg")
```

EnImpute can be run on Windows, MacOS and UNIX platforms. The main function of the package is EnImpute. To run the function EnImpute, one can use the following code after loading the package:

Usage

```
EnImpute(count, scale.factor = 10000, trim = 0.3, ALRA = TRUE,
  DCA = TRUE, DrImpute = TRUE, MAGIC = TRUE, SAVER = TRUE,
  scImpute = TRUE, scRMD = TRUE, Seurat = TRUE, ALRA.k = 0,
  ALRA.q = 10, DCA.normtype = "zheng", DCA.type = "zinb-conddisp",
  DCA.l2 = 0, DCA.l1 = 0, DCA.l2enc = 0, DCA.l1enc = 0, DCA.ridge = 0,
```

```
  DCA.gradclip = 5, DCA.activation = "relu", DCA.hiddensize = "64,32,64",
  DCA.hyper = FALSE, DCA.hypern = 1000, DrImpute.ks = 10:15,
  DrImpute.dists = c("spearman", "pearson"), DrImpute.method = "mean",
  DrImpute.cls = NULL, MAGIC.k = 10, MAGIC.alpha = 15, MAGIC.t = "auto",
  MAGIC.npca = 20, MAGIC.t.max = 20, MAGIC.knn.dist.method = "euclidean",
  MAGIC.n.jobs = 1, SAVER.do.fast = TRUE, SAVER.ncores = 2,
  SAVER.size.factor = NULL, SAVER.npred = NULL, SAVER.null.model = FALSE,
  SAVER.mu = NULL, scImpute.drop_thre = 0.5, scImpute.Kcluster = 5,
  scImpute.labeled = FALSE, scImpute.labels = NULL,
  scImpute.genelen = NULL, scImpute.ncores = 1, scRMD.tau = NULL,
  scRMD.lambda = NULL, scRMD.candidate = 0.05, Seurat.genes.use = NULL,
  Seurat.genes.fit = NULL, Seurat.gram = TRUE)
```

The arguments are listed as follows:

Arguments

`count`	raw read count matrix. The rows correspond to genes and the columns correspond to cells.
`scale.factor`	scale factor used to re-scale the imputed results generated by different individual methods. Default is 10000.
`trim`	specifies the fraction (between 0 and 0.5) of observations to be trimmed from each end before the mean is computed. Default is 0.3.
`ALRA`	a boolean variable that defines whether to impute the raw data using the ALRA method. Default is TRUE.
`DCA`	a boolean variable that defines whether to impute the raw data using the DCA method. Default is TRUE.
`DrImpute`	a boolean variable that defines whether to impute the raw data using the DrImpute method. Default is TRUE.
`MAGIC`	a boolean variable that defines whether to impute the raw data using the MAGIC method. Default is TRUE.
`SAVER`	a boolean variable that defines whether to impute the raw data using the SAVER method. Default is TRUE.
`scImpute`	a boolean variable that defines whether to impute the raw data using the scImpute method. Default is TRUE.
`scRMD`	a boolean variable that defines whether to impute the raw data using the scRMD method. Default is TRUE.
`Seurat`	a boolean variable that defines whether to impute the raw data using the Seurat method. Default is TRUE.
`ALRA.k`	the rank of the rank-k approximation in ALRA. Set to 0 for automated choice of k. Default is 0.

`ALRA.q` the number of power iterations in randomized SVD used by ALRA. Default is 10.

`DCA.normtype` a string variable specifying the type of size factor estimation in DCA. Possible values: "deseq", "zheng". Default is "zheng".

`DCA.type` a string variable specifying type of autoencoder in DCA. Possible values: "normal", "poisson", "nb", "nb-shared", "nb-conddisp", "nb-fork", "zinb", "zinb-shared", "zinb-conddisp", "zinb-fork". Default is "zinb-conddisp".

`DCA.l2` a real number specifying the L2 regularization coefficient in DCA. Default is 0.

`DCA.l1` a real number specifying the L1 regularization coefficient in DCA. Default is 0.

`DCA.l2enc` a real number specifying the encoder-specific L2 regularization coefficient in DCA. Default is 0.

`DCA.l1enc` a real number specifying the encoder-specific L1 regularization coefficient in DCA. Default is 0.

`DCA.ridge` a real number specifying the L2 regularization coefficient for dropout probabilities in DCA. Default is 0.

`DCA.gradclip` a real number specifying the Clip grad values in DCA. Default is 5.

`DCA.activation` a string value specifying the activation function of hidden unit in DCA. Default is "relu".

`DCA.hiddensize` a string vector specifying the size of hidden layers in DCA. Default is "64,32,64".

`DCA.hyper` a logical value specifying whether hyperparameter search is performed in DCA.

`DCA.hypern` an integer specifying the number of samples drawn from hyperparameter distributions during optimization in DCA. Default is 1000.

`DrImpute.ks` an integer vector specifying the number of cell clustering groups in DrImpute. Default is 10:15.

`DrImpute.dists` a string vector specifying the distance metrics in DrImpute. Default is c("spearman", "pearson").

`DrImpute.method` a string specifying the method used for imputation in DrImpute. Use "mean" for mean imputation, "med" for median imputation.

`DrImpute.cls` a matrix specifying the clustering information manually provided by users in DrImpute. The rows represent different clusterings, and the columns represent cells. Default is NULL, which means the user do not provide the clustering information.

`MAGIC.k` an integer specifying the number of nearest neighbors on which to build kernel in MAGIC. Default is 10.

`MAGIC.alpha` an integer specifying the decay rate of kernel tails in MAGIC. Default is 15.

`MAGIC.t` an integer specifying the diffusion time for the Markov Affinity Matrix in MAGIC. Default is "auto". For detail about the approach to set paramter t automatically, please refer to the reference.

`MAGIC.npca` an integer specifying the number of PCA components in MAGIC. Default is 20.

`MAGIC.t.max` an integer specifying the maximum value of t to test for automatic t selection in MAGIC. Default is 20.

`MAGIC.knn.dist.method`
a string value specifying the metric for building kNN graph in MAGIC. Recommended values: "euclidean", "cosine". Default is "euclidean".

`MAGIC.n.jobs` an integer specifying the number of jobs used for computation in MAGIC. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all. For n.jobs below -1, (n.cpus + 1 + n.jobs) are used. Thus for n.jobs = -2, all CPUs but one are used.

`SAVER.do.fast`
a boolean variable specifying whether the prediction step is approximated in SAVER. Default is TRUE.

`SAVER.ncores` number of cores to use in SAVER. Default is 1.

`SAVER.size.factor`
a vector of cell size specifying the normalization factors in SAVER. If the data is already normalized or normalization is not desired, set size.factor = 1. Default uses mean library size normalization.

`SAVER.npred` number of genes for regression prediction in SAVER. Selects the top npred genes in terms of mean expression for regression prediction. Default is all genes.

`SAVER.null.model`
a boolean variable specifying whether to use mean gene expression as prediction in SAVER. Default is FALSE

`SAVER.mu` matrix of prior means in SAVER.

`scImpute.drop_thre`
a number (between 0 and 1) specifying the threshold on dropout probability in scImpute. Default is 0.5.

`scImpute.Kcluster`
an integer specifying the number of cell subpopulations in scImpute. Default is 10.

`scImpute.labeled`
a boolean variable indicating whether cell type information is given in scImpute. Default is FALSE.

`scImpute.labels`
a character vector specifying the cell type in scImpute. Only needed when `labeled = TRUE`. Default is NULL

`scImpute.genelen`
an integer vector giving the length of each gene in scImpute. Default is NULL.

`scImpute.ncores`
an integer specifying the number of cores used for parallel computation in scImpute. Default is 1.

`scRMD.tau` a non-negative real number specifying the tuning parameter to penalize the sparse term. Default is NULL.

`scRMD.lambda` a non-negative real number specifying the tuning parameter to penalize the row rank term. Default is NULL.

`scRMD.candidate`
a real number (0 to 1) specifying the cutoff for candidate drop out. Default is 0.05.

`Seurat.genes.use`
a vector of genes that can be used for building the models in Seurat. Default use the high variable gene detected by the `FindVariableGenes` in the Seurat package.

`Seurat.genes.fit`
a vector of genes to impute values for. Default is all genes

`Seurat.gram` a logical value specifying whether the Gram matrix is precomputed in Seurat. Default is TRUE.

`Seurat.genes.use`
a vector of genes that can be used for building the models in Seurat. Default use the high variable gene detected by the `FindVariableGenes` in the Seurat package.

`Seurat.genes.fit`
a vector of genes to impute values for. Default is all genes.

The output of `EnImpute` is a list with the following components:

Value

`count.EnImpute.log`
Imputed count matrix generated by EnImpute (log scale).

`count.EnImpute.exp`
Imputed count matrix generated by EnImpute (exp scale).

`count.imputed.individual`
Imputed count matrices generated by different individual imputation methods (exp scale).

`Methods.used` The individual methods used by EnImpute.

For detailed usages and real data application examples, please refer to "EnImpute-manual.pdf" and directory containing scripts for performing real data analysis, which are available at https://github.com/Zhangxf-ccnu/EnImpute. Any bugs, suggestions and request related to EnImpute can be reported through the GitHub repository.

3 Shiny web application

The Shiny application interface is divided into three panels (Figure S1). In the left panel, users can upload the raw count matrix according to the data input box. The count matrix needs to be a .csv file containing a $genes \times cells$ matrix in which the rows denote genes and the columns represent cells. The data file is required to contain the names of the cells as its first line, and contain the names of the genes as its first column. Examples are provided in the directory "data". The scale factor used to re-scale the imputed results generated by different individual imputation methods and the trim parameter for computing trimmed mean can be set in the left panel. Users can choose the individual imputation methods and set their parameters in the left panel. When imputing is finished, t-SNE visualization of

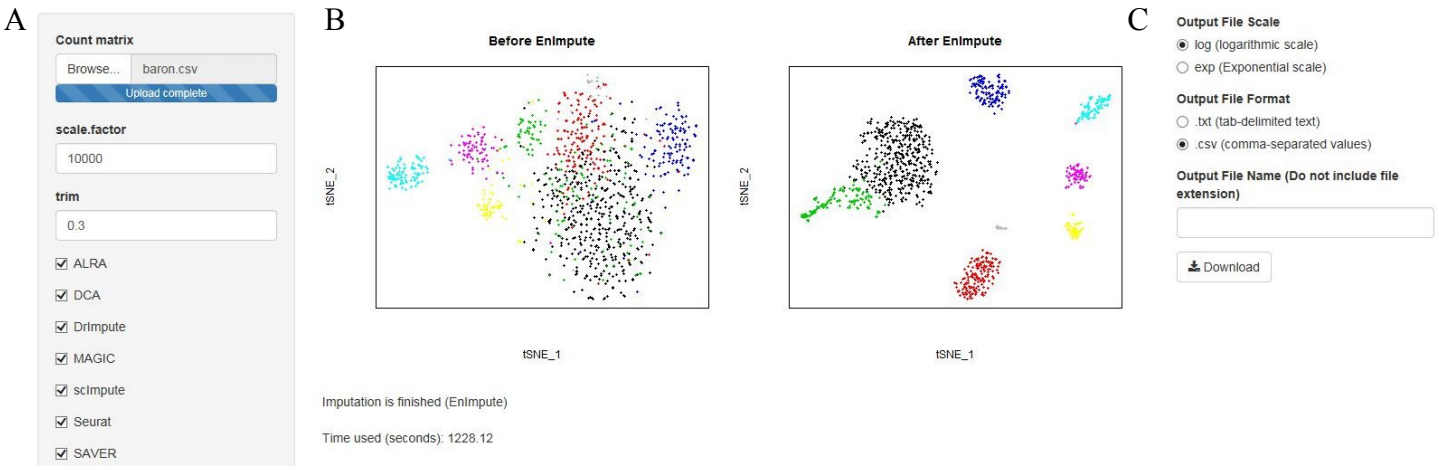

Figure S1: Interface of EnImpute Shiny application. The interface is divided into three panels. (A) Users can upload the data and set the tuning parameters in the left panel. (B) t-SNE visualization of the raw data and the data imputed by EnImpute are shown in the middle panel. (C) Users can download the imputed data according to the right panel.

the raw data and the data imputed by EnImpute will be shown in the middle panel, color-coded by the clusters identified from the imputed data. The time used to run EnImpute is also provided in the middle panel. Users can download the imputed data with the download button in the right panel.

4 Assessing the performance through down-sampling experiments

Given that it is difficult to obtain the gold standard of the true expression levels, we investigate imputation accuracy of EnImpute through the down-sampling experiments following the method of [5].

4.1 Datasets

We consider the four datasets used in [5]: Baron [9], Chen [10], Manno [11], and Zeisel [12]. To generate the reference datasets for real data analysis, high-quality cells and genes with high expression from the original datasets are selected by Huang et al to treat as the true expression [5]. To mimic dropout events, downsampled (observed) datasets are generated by drawing from a Poisson distribution with cell-specific efficiency loss. The reference data and observed data for the four datasets are downloaded from https://github.com/mohuangx/SAVER-paper/tree/master/SAVER-data. The statistics of the four datasets are presented in Table S2.

4.2 EnImpute improves the recovery of the true expression levels

To evaluate the performance of different methods on recovering the true expression levels, we run the eight individual imputation methods (ALRA, DCA, DrImpute, MAGIC, SAVER, scImpute, scRMD and Seurat) and our EnImpute on each observed dataset. We do not run DrImpute on the chen dataset due to the memory limit on a desktop computer with 8GB RAM. The individual imputation methods are run with default settings in the R package EnImpute. For EnImpute, we set the parameter $trim = 0.3$. The

Table S2: Statistics of the datasets used in the down-sampling experiments

Datasets	Number of cells	Number of genes	Percentage of zeros in reference (observed) data
Baron	1,076	2,284	46.5% (87.5%)
Chen	7,712	2,159	54.0% (90.5%)
Manno	947	2,059	39.7% (84.4%)
Zeisel	1,799	3,529	27.3% (86.1%)

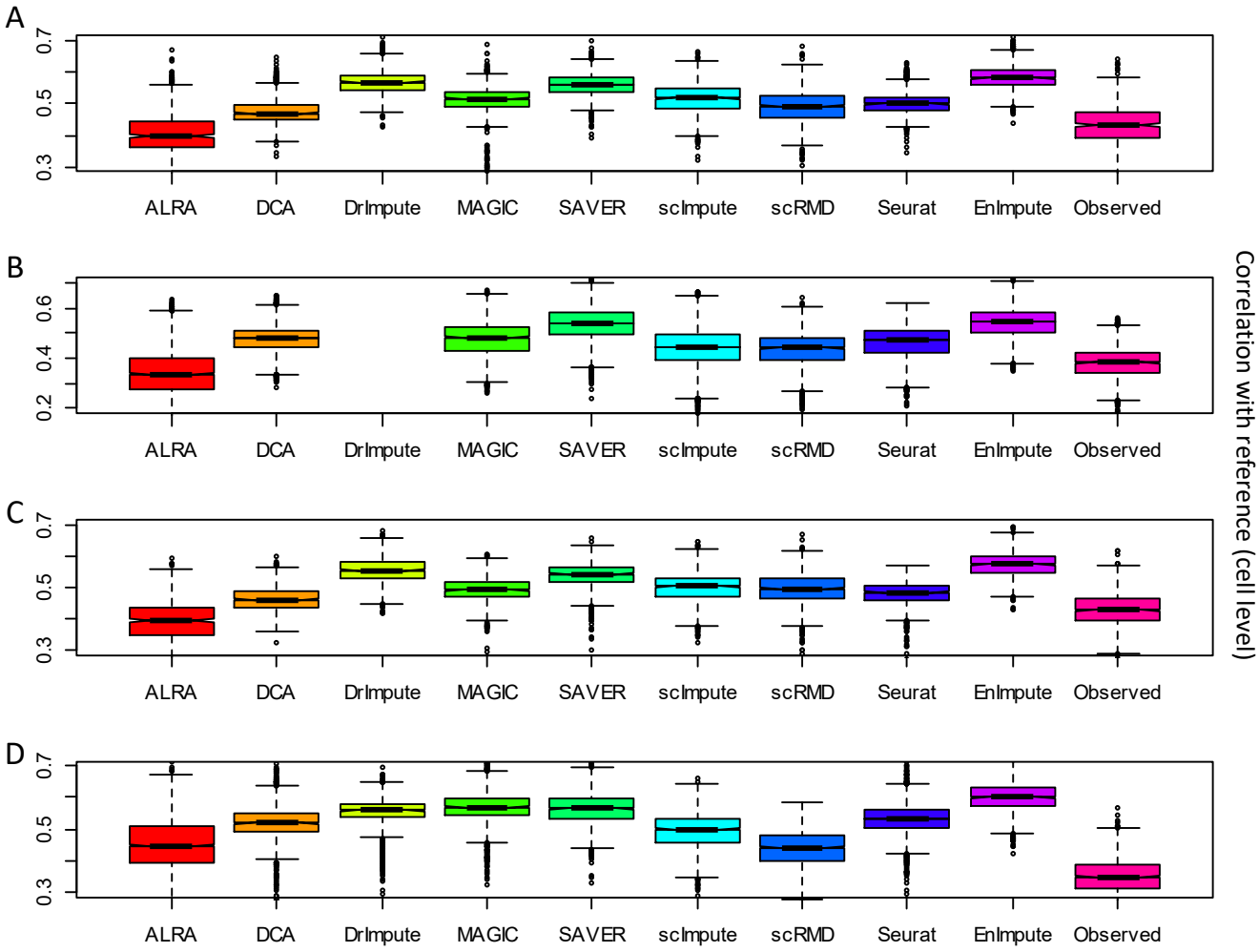

Figure S2: Performance of different methods measured by cell-wise correlation with the reference data. For each plot, the Y-axis represents the Pearson cell-wise correlation across genes between the reference and imputed data (as well as the observed data). (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

Pearson gene-wise correlation across cells and the Pearson cell-wise correlation across genes between the reference and the data imputed by different methods, as well as between the reference and observed data, are calculated. As can be seen from Figure S2, all methods (except ALRA) perform better than the observed data in terms of cell-wise correlation with reference data. However, only DrImpute, SAVER, scRMD, and EnImpute outperform the observed data on all the four datasets in terms of gene-wise correlation with reference (Figure S3). In addition, our EnImpute performs better than all compared methods on all datasets in terms of both cell-wise correlation and gene-wise correlation with reference.

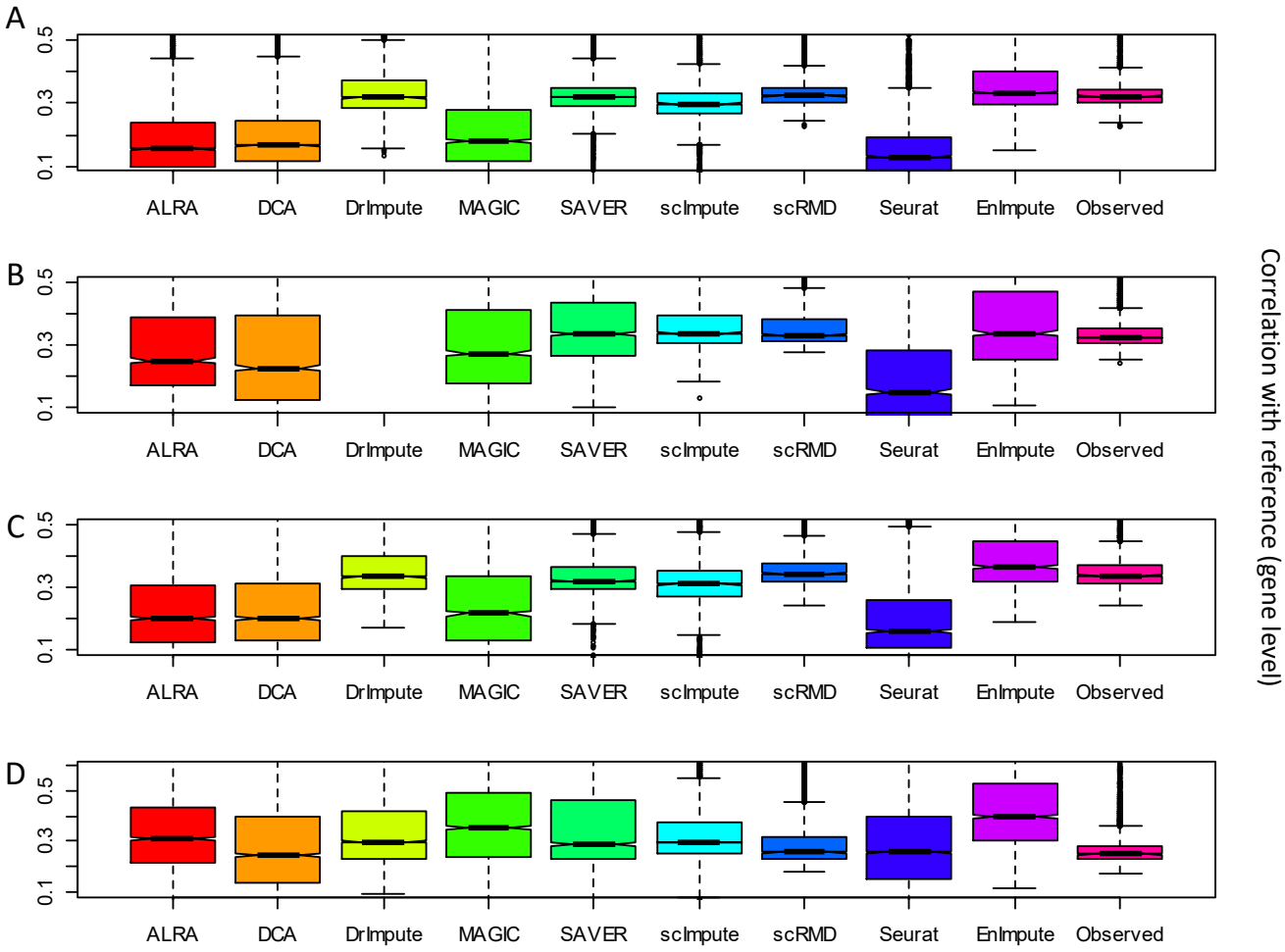

Figure S3: Performance of different methods measured by gene-wise correlation with the reference data. For each plot, the Y-axis represents the Pearson gene-wise correlation across cells between the reference and imputed data (as well as the observed data). (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

4.3 EnImpute improves the recovery of cell-to-cell and gene-to-gene correlations

Cell-to-cell correlation matrices are important for the identification of cell-types, and gene-to-gene correlation matrices are crucial for the reconstruction of gene networks. Therefore, we evaluate the performance of different imputation methods in terms of the recovery of cell-to-cell and gene-to-gene correlation matrices. Following [5], we define the correlation matrix distance (CMD) between two correlation matrices, R_1 and R_2, as $d(R_1, R_2) = 1 - \frac{tr(R_1 R_2)}{\|R_1\|_F \|R_2\|_F}$. The CMD measure ranges from 0 (equal) to 1 (maximum difference). The CMD between the Pearson correlation matrix derived from the imputed (and observed) matrix and the Pearson correlation matrix derived from the reference matrix is computed. As shown in Figures S4 and S5, EnImpute has competitive performance in the recovery of cell-to-cell and gene-to-gene correlation matrices on all the four datasets.

4.4 EnImpute improves cell clustering and visualization

We also investigate the effect of different imputation methods on cell clustering and visualization. The R package Seurat [13] is used to carry out cell clustering and t-SNE visualization. We perform cell clustering and t-SNE visualization on the reference, observed and imputed data. The cell clusters generated

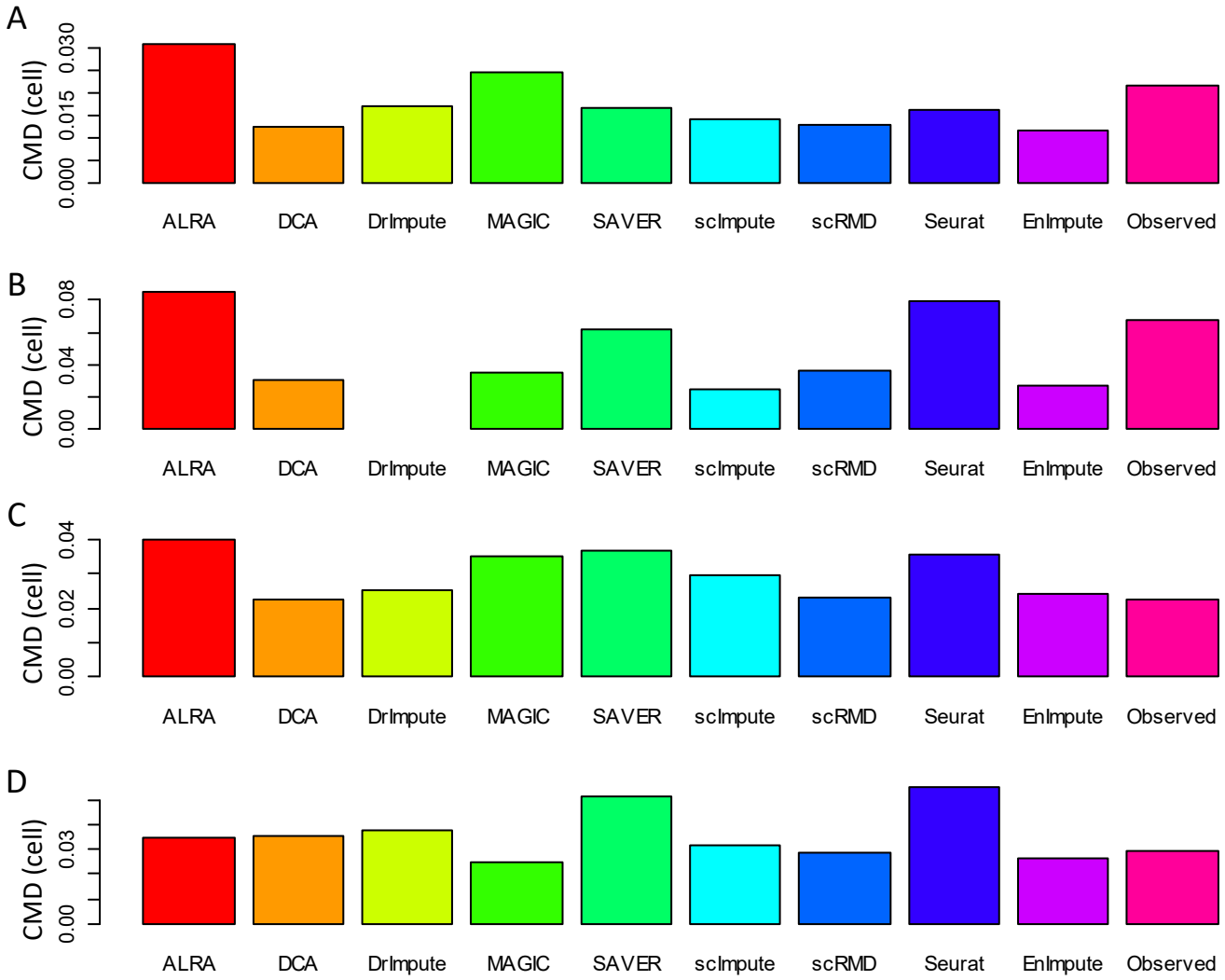

Figure S4: Performance of different methods measured by the recovery of cell-to-cell correlations, as measured by correlation matrix distance. For each plot, the Y-axis represents the correlation matrix distance between the cell-to-cell Pearson correlation matrix derived from the imputed (and observed) matrix and that derived from the reference matrix. (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

from the reference data are treated as the ground truth. The clustering accuracy on the observed and imputed data is assessed in terms of adjusted Rand index (ARI) (computed using the function adjustedRandIndex in the R package mclust) and t-SNE visualization. A higher ARI score indicates a better result. EnImpute achieves competitive ARI scores on all the four datasets (Figure S6). The t-SNE plots also show that the data imputed by EnImpute have the best representation of the clusters identified from the reference data (Figures S7, S8, S9, and S10).

In summary, all methods (except ALRA) improve cell-wise correlation across all datasets, and DrImpute, SAVER, scRMD and EnImpute improve gene-wise correlation across all datasets. DCA, DrImpute, scImpute, scRMD and EnImpute perform better than the observed datasets in terms of recovery of cell-to-cell correlation, and ALRA, DrImpute and EnImpute improve the recovery of gene-to-gene correlation across all datasets. Most imputation methods have better performance than the observed datasets on capturing the clusters identified from the reference data. Due to the fact that each individual imputation method is developed to capture one aspect of the data, the performance of the individual

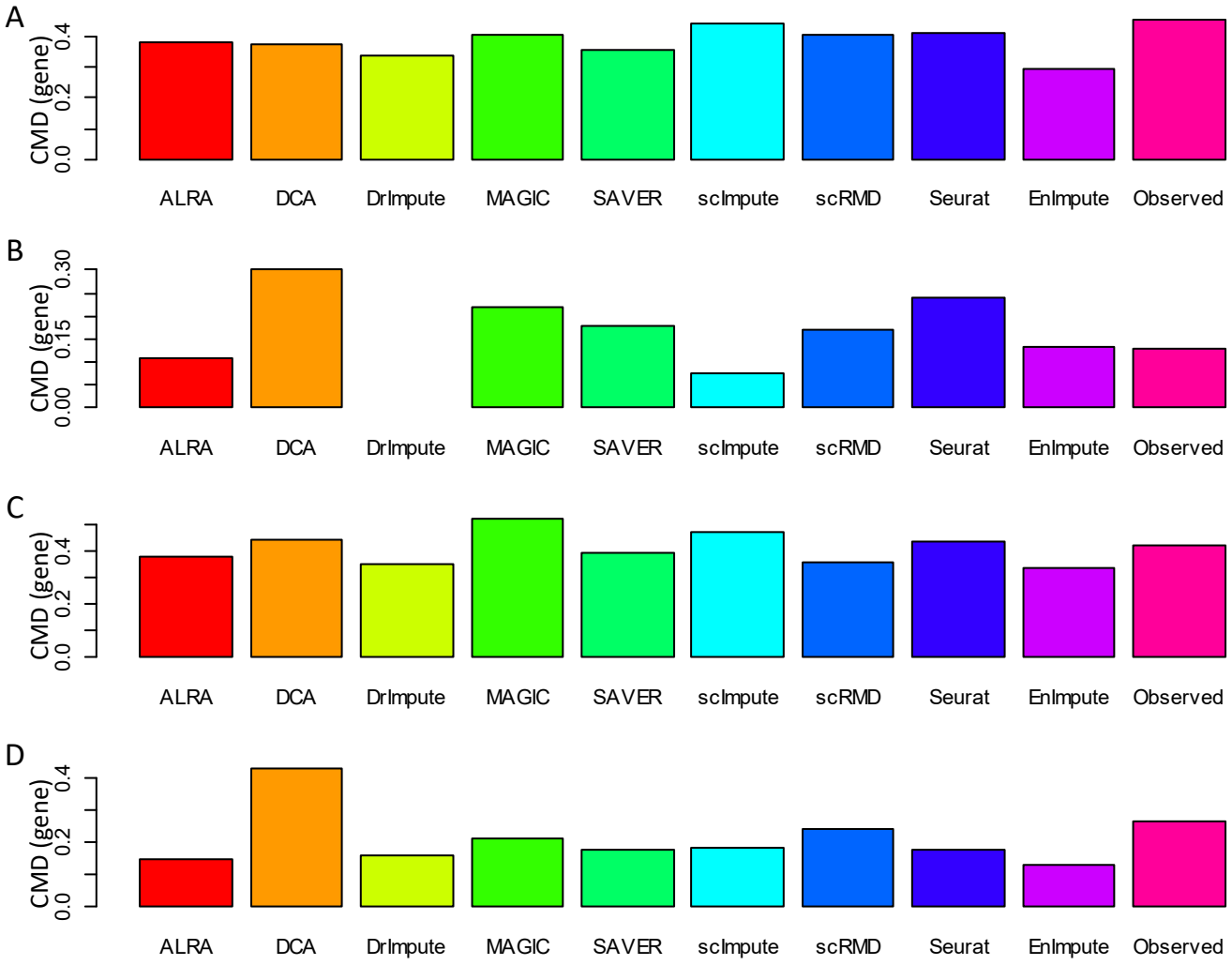

Figure S5: Performance of different methods measured by the recovery of gene-to-gene correlations, as measured by correlation matrix distance. For each plot, the Y-axis represents the correlation matrix distance between the gene-to-gene Pearson correlation matrix derived from the imputed (and observed) matrix and that derived from the reference matrix. (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

methods may differ substantially with respect to the datasets and the evaluation methods. Our EnImpute that combines the results from multiple individual imputation methods has competitive performance on all datasets using all evaluation methods.

5 Assessing the performance through differential expression analysis

In this section, we investigate the effect of imputation on differential expression analysis. We conduct the experiments in a similar way to [2, 6, 7]. The differential expression analysis results between the bulk and scRNA-seq data from the same experiment are compared. A real dataset that includes both bulk and scRNA-seq experiments on human embryonic stem cells (ESC) and definitive endoderm cells (DEC) is used [14]. This dataset includes six samples of bulk RNA-seq (four H1 ESC samples and two DEC samples) and 350 samples of scRNA-seq (212 H1 ESC samples and 138 DEC samples). Here

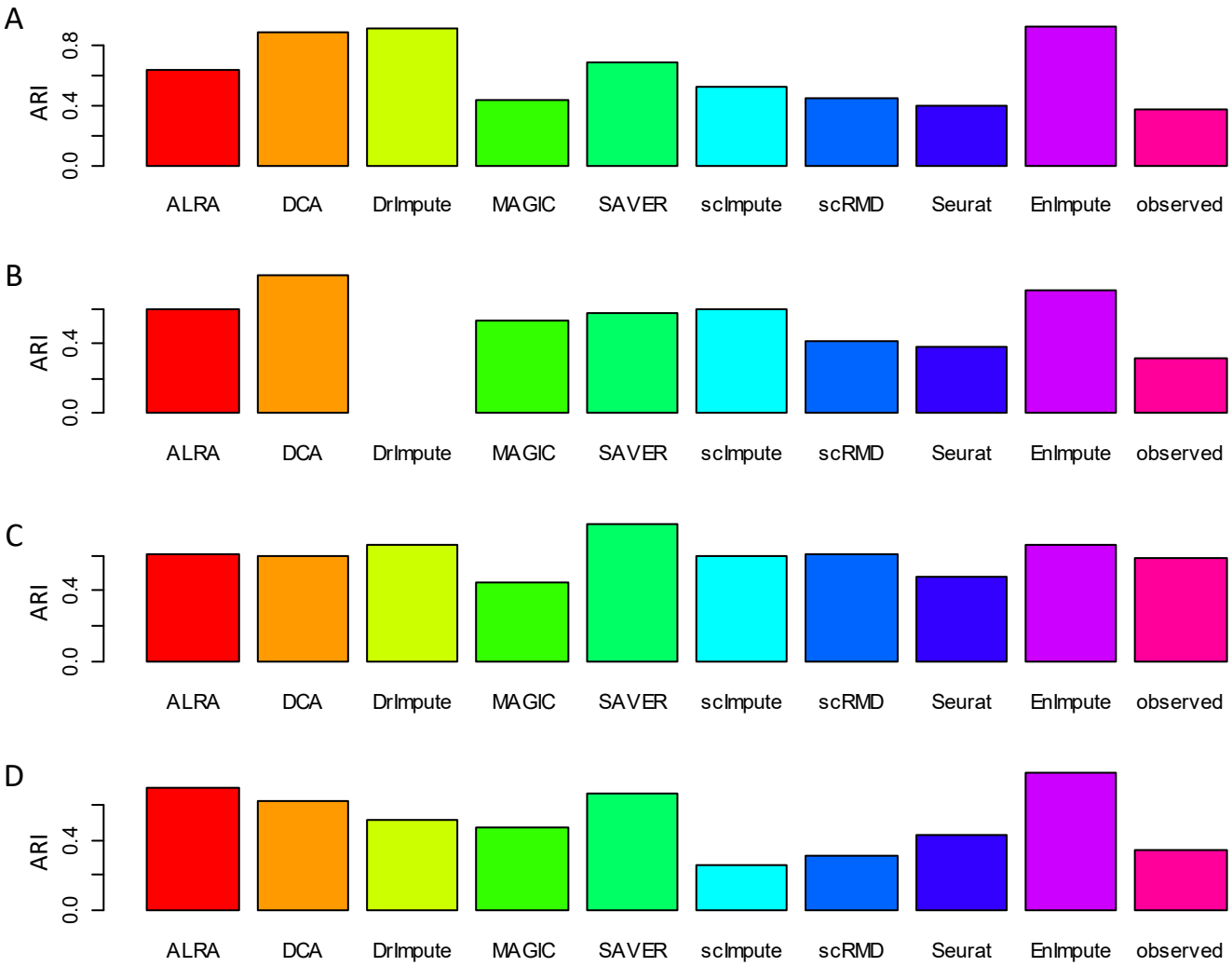

Figure S6: Comparison of the cell clusters generated from the imputed (and observed) data with the reference data. For each plot, the Y-axis represents the adjusted Rand index between the clusters generated from the reference data and that generated from the imputed (and observed) data. (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

we only consider the 1,018 highly variable genes identified from the scRNA-seq data by the function FindVariableGenes in the R package Seurat (parameters are set with default values).

We apply the eight individual methods and our EnImpute to impute the scRNA-seq data. Differential expression analysis is performed on the observed raw data and the data imputed by each method, respectively. The R package edgeR [15] is used to identify differentially expressed genes. Parameters are set as the default values. We choose the top 200 and 400 genes ranked by p-values from the bulk data as the gold standard. We then choose top k differential expressed genes ranked by p-values from the scRNA-seq data and compare them with the gold standard. We find that EnImpute outperforms the individual imputation methods as well as the observed raw data (Figure S11). Since the overlap analysis results depend on the choice of the gold standard, we also directly calculate the Spearman correlation coefficient between the p-values derived from the scRNA-seq data and those derived from the bulk data. Figure S12A shows that EnImpute obtains the highest Spearman correlation coefficient. We also use a bootstrapping approach to test the robustness of the results following the method of [2]. We repeat differential expression analysis 100 times. At each repetition, 50% cells are randomly sampled and differential expression analysis is performed on the sampled data, then the Spearman correlation between

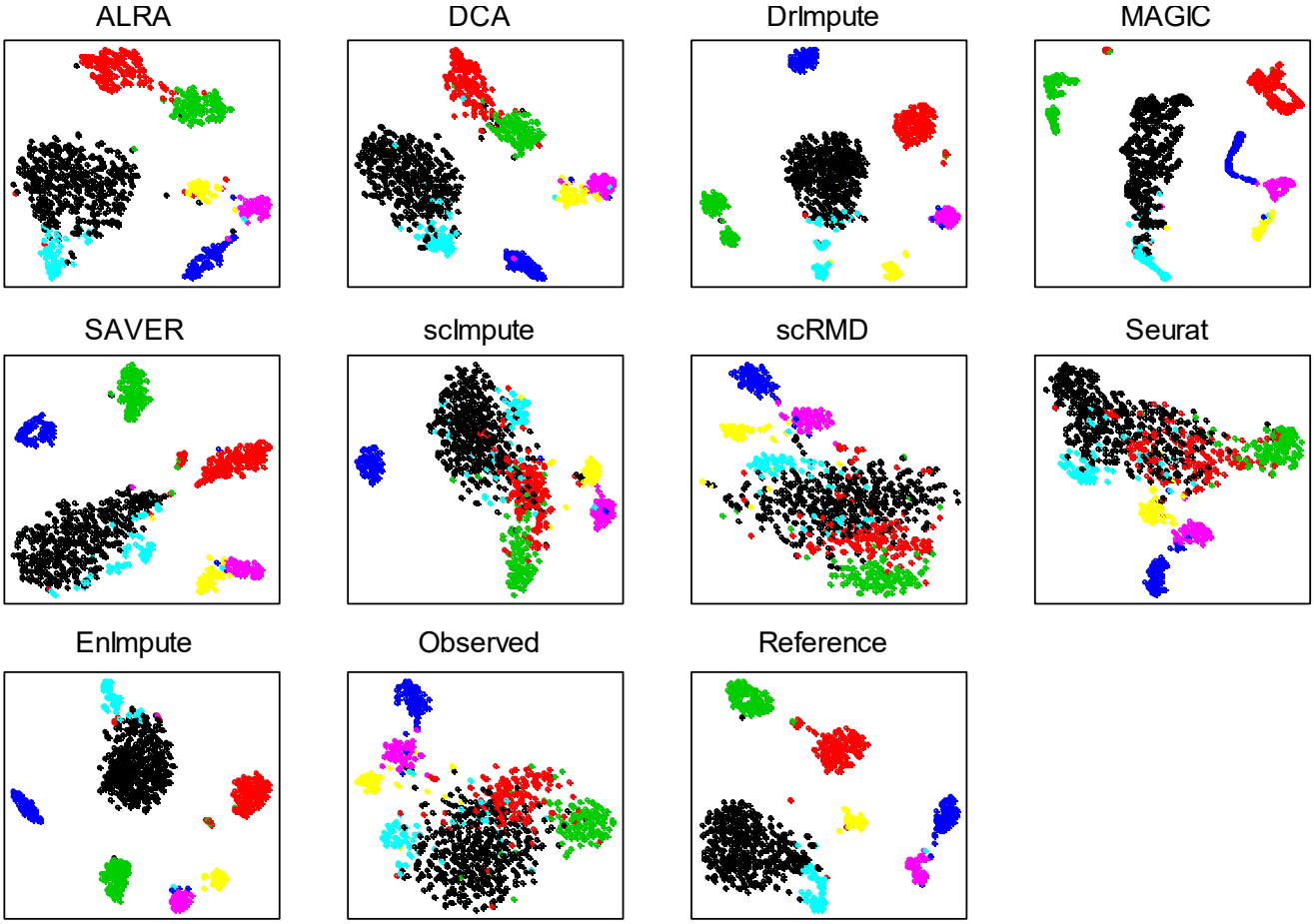

Figure S7: t-SNE visualization of the reference, observed and imputed datasets from Baron et al [9]. The cells are color-coded by the clusters identified from the reference data.

the p-values derived from the sampled single-cell data and bulk data are calculated. As can be seen from Figure S12B, EnImpute achieves the highest correspondence with the bulk data. These results reveal that EnImpute performs best in increasing the agreement between bulk and single-cell differential expression analysis.

6 Assessing the performance through clustering and visualization analysis

We assess the performance of different imputation methods for cell clustering and visualization. We consider three datasets: the Camp dataset (GSE75140) [16], the Lake dataset (http://genome-tech.ucsd.edu/public/Lake_Science_2016/) [17], and the Usoskin dataset (GSE59739) [18]. The Camp dataset includes 734 single-cell transcriptomes from human fetal neocortex or human cerebral organoids. After removing cells with type "Unknown", the Camp dataset consisting of 553 cells from 5 different cell types. The Lake dataset consisting of 3,042 cells from 16 different cell types. The Usoskin dataset includes 622 cells from 4 different cell types. For all the three datasets, we only consider the high variable genes. The function FindVariableGenes in the R package Seurat with default parameter settings is used to identify the high variable genes. This leads to 1,560, 1,313 and 1,339 genes for the three datasets, respectively.

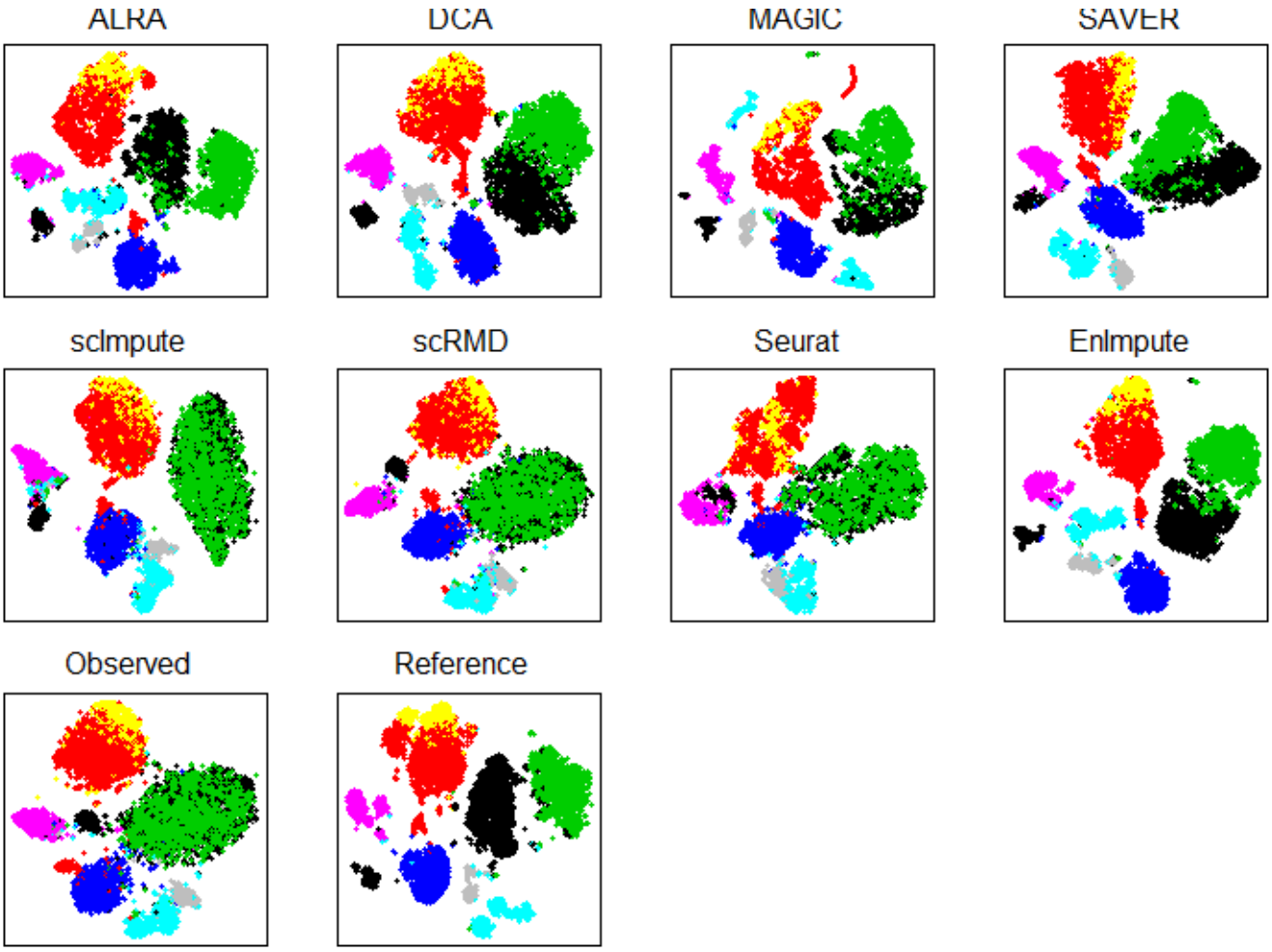

Figure S8: t-SNE visualization of the reference, observed and imputed datasets from Chen et al [10]. The cells are color-coded by the clusters identified from the reference data.

We use three methods to carry out clustering analysis: SC3 [19], tSNE+kmeans, and Seurat [13]. The R package SC3 is used to implement SC3. For tSNE+kmeans, the RunTSNE function in the R package Seurat is used to reduce the data dimension (the dimension is set to 2), and the kmeans function in the package stats is followed to cluster the cells. For SC3 and tSNE+kmeans, the number of clusters is set to the number of known cell types. For Seurat, the function FindClusters in the R package Seurat is used to find clusters, where the resolution parameter is set to 1.

We first apply the imputation methods to the three raw observed datasets, and then run the three clustering methods on the observed and imputed data. The adjusted Rand index (ARI) is used to assess the clustering performance. The performance of different imputation methods depends on the used clustering methods (Figure S13). For example, MAGIC has competitive performance when tSNE+kmeans is used. Overall, EnImpute performs much better than the other imputation methods according to all the three clustering algorithms. To further illustrate the performance of different imputation methods, we plot the two dimensional t-SNE results for the raw observed data and the imputed data (Figure S14). The cells are color-coded by the known cell types. The cell types are not well separated by the observed data. EnImpute provides more clearer separation of cell types than individual imputation methods. These results indicate that EnImpute can improve the identification of cell types.

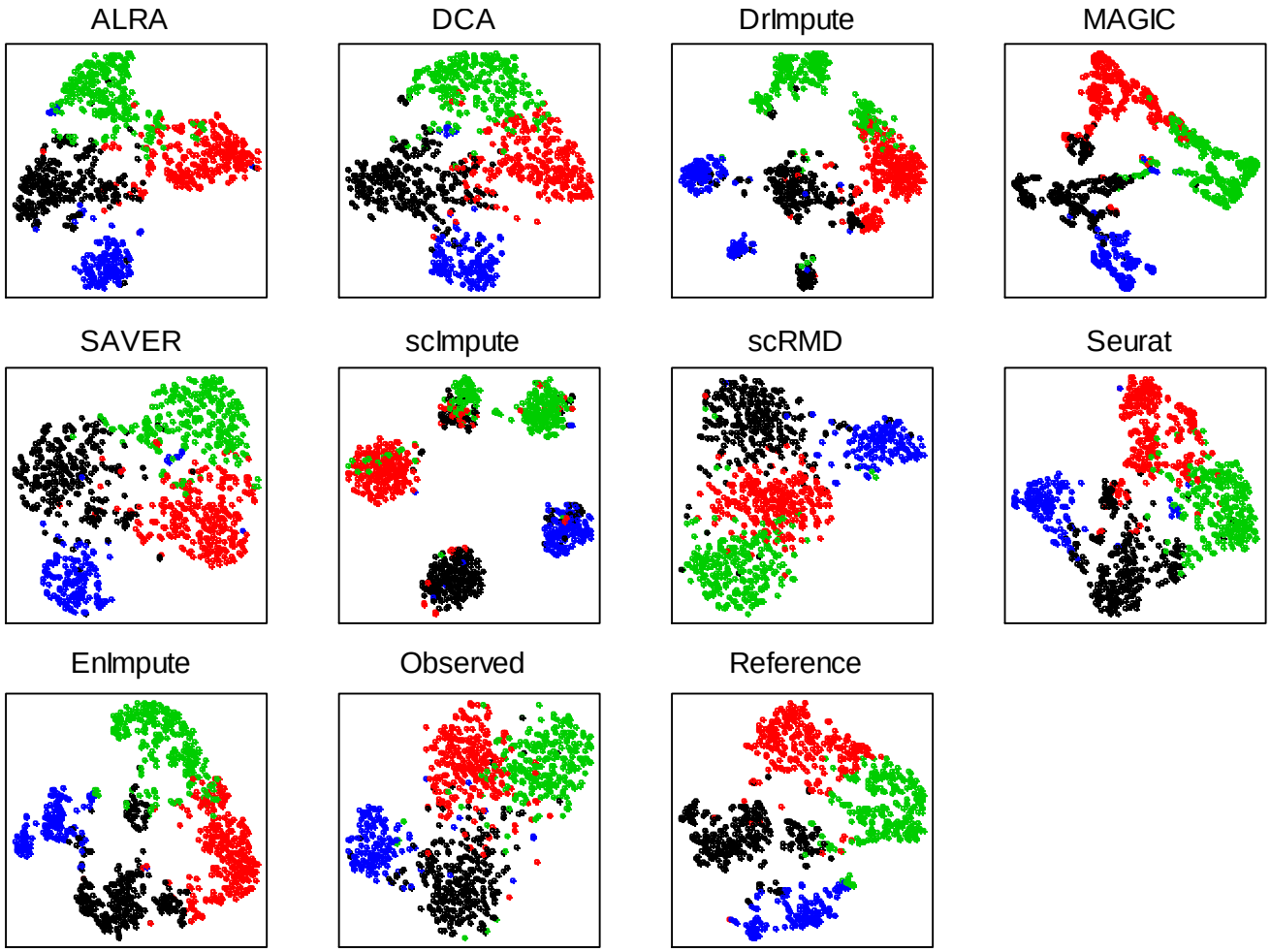

Figure S9: t-SNE visualization of the reference, observed and imputed datasets from Manno et al [11]. The cells are color-coded by the clusters identified from the reference data.

7 Sensitivity analysis of parameters

In EnImpute, there is a parameter $trim$ that specifies the fraction (0 to 0.5) of observations to be trimmed from each end before calculating the mean. If $trim = 0$, the arithmetic mean will be used, and if $trim = 0.5$, the sample median will be computed. In the above experiments, we set $trim = 0.3$ as default value. In this section, we analyze the effect of $trim$ on the performance of EnImpute through down-sampling experiments. The experiments are conducted using the four datasets (Baron, Chen, Manno, and Zeisel) used in Section S4. We run EnImpute with different values of $trim$ $(0, 0.1, \ldots, 0.5)$ on the down-sampled observed data, and evaluate the imputed results in terms of the recovery of the true expression levels, and the recovery of cell-to-cell and gene-to-gene correlations.

From Figures S15, S16, S17, and S18, we observe that the performance of EnImpute is not very sensitive to the values of $trim$. However, EnImpute often obtains the best performance when $trim = 0.2, 0.3$. This may be owing to the fact the trimmed mean is less sensitive to outliers than the standard mean ($trim = 0$) and can use more information from the observations than the median ($trim = 0.5$).

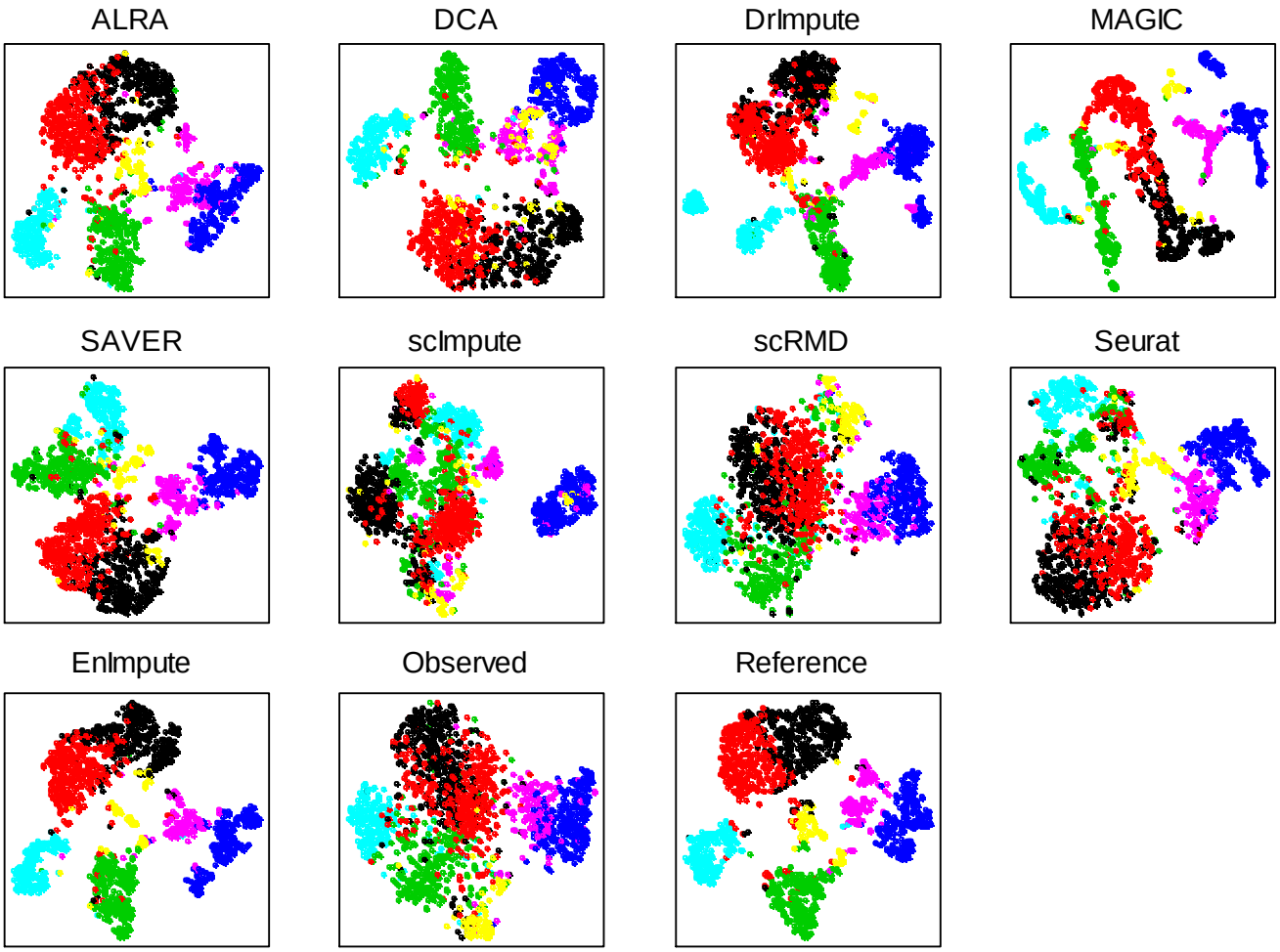

Figure S10: t-SNE visualization of the reference, observed and imputed datasets from Zeisel et al [12]. The cells are color-coded by the clusters identified from the reference data.

References

[1] George C. Linderman et al. Zero-preserving imputation of scrna-seq data using low-rank approximation. *bioRxiv*, 2018.

[2] Gökcen Eraslan et al. Single cell rna-seq denoising using a deep count autoencoder. *bioRxiv*, page 300681, 2018.

[3] Il-Youp Kwak et al. Drimpute: Imputing dropout events in single cell rna sequencing data. *BMC Bioinformatics*, 19:220, 2018.

[4] David van Dijk et al. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 174:1–14, 2018.

[5] Mo Huang et al. Saver: gene expression recovery for single-cell rna sequencing. *Nature Methods*, 15:539–542, 2018.

[6] Wei Vivian Li and Jingyi Jessica Li. An accurate and robust imputation method scimpute for single-cell rna-seq data. *Nat Commun*, 9(1):997, 2018.

[7] Chong Chen, Changjing Wu, Linjie Wu, Yishu Wang, Minghua Deng, and Ruibin Xi. scrmd:

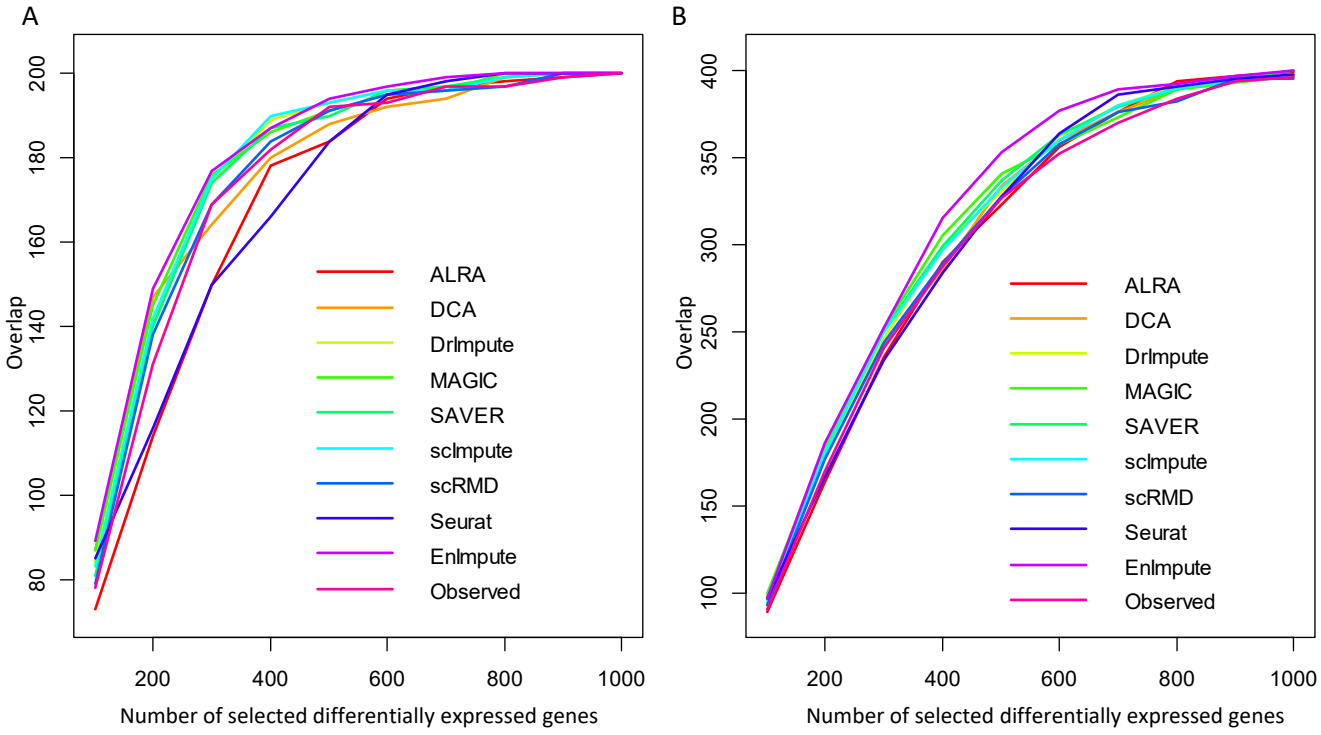

Figure S11: The overlap of the differential expression genes detected from the scRNA-seq data (including the observed raw data and the data imputed by different methods) with the golden standard gene sets. The golden standard gene sets are chosen as the top 200 (A) and 400 (B) differential expressed genes detected from the bulk data. The X-axis represents the number of differential expression genes selected from scRNA-seq data, and the Y-axis represents the overlap between the differential expression genes selected from the scRNA-seq data and the golden standard gene sets.

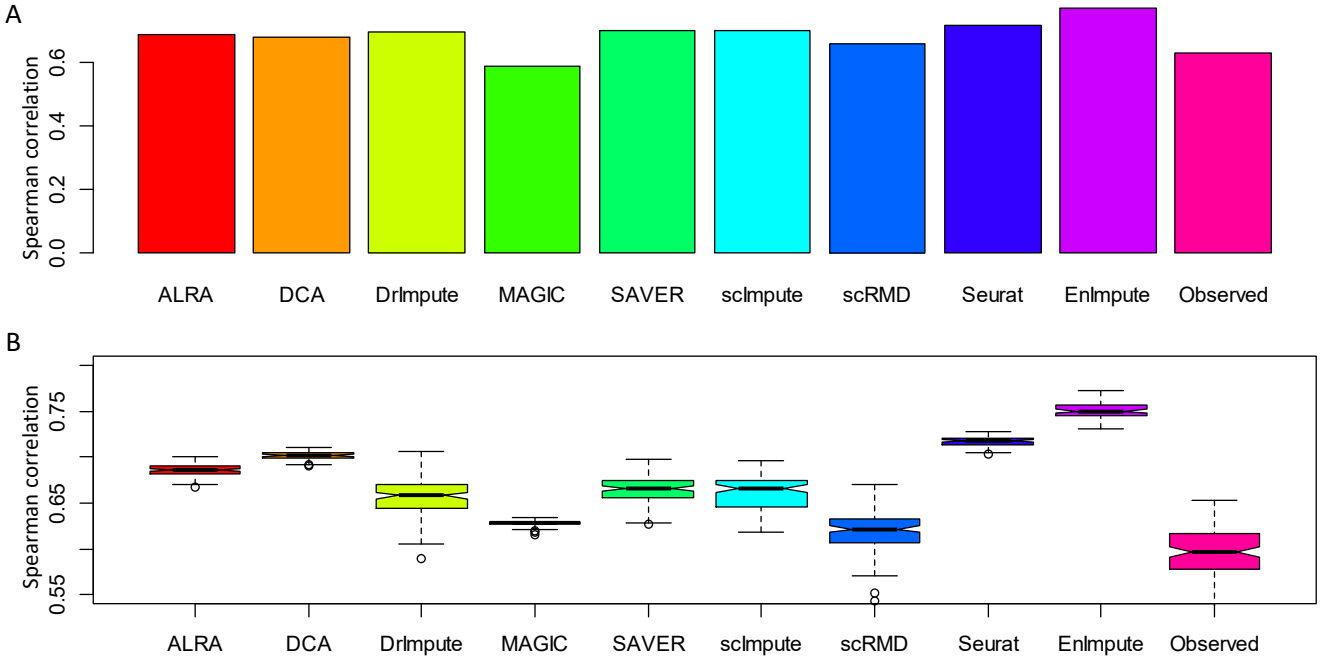

Figure S12: The agreement between single-cell and bulk differential expression analysis. (A) The Spearman correlation between the p-values derived from the scRNA-seq data (including the observed raw data and the imputed data) and those derived from the bulk data. (B) The distribution of the Spearman correlations obtained from bootstrapping differential expression analysis.

Imputation for single cell rna-seq data via robust matrix decomposition. *bioRxiv*, page 459404, 2018.

[8] Rahul Satija et al. Spatial reconstruction of single-cell gene expression data. *Nat. Biotechnol.*, 33(5):495–502, 2015.

[9] Maayan Baron et al. A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure. *Cell Systems*, 3(4):346–360, 2016.

[10] Renchao Chen, Xiaoji Wu, Lan Jiang, and Yi Zhang. Single-cell rna-seq reveals hypothalamic cell diversity. *Cell Reports*, 18(13):3227–3241, 2017.

[11] Gioele La Manno et al. Molecular diversity of midbrain development in mouse, human, and stem cells. *Cell*, 167(2):566–580, 2016.

[12] Amit Zeisel et al. Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142, 2015.

[13] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36(5):411–420, 2018.

[14] Lifang Chu, Ning Leng, Jue Zhang, Zhonggang Hou, Daniel Mamott, David T Vereide, Jeea Choi, Christina Kendziorski, Ron Stewart, and James A Thomson. Single-cell rna-seq reveals novel regulators of human embryonic stem cell differentiation to definitive endoderm. *Genome Biology*, 17(1):173, 2016.

[15] Mark D Robinson, Davis J Mccarthy, and Gordon K Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010.

[16] J Gray Camp, Farhath Badsha, Marta Florio, Sabina Kanton, Tobias Gerber, Michaela Wilschbrauninger, Eric Lewitus, Alex M Sykes, Wulf Hevers, Madeline A Lancaster, et al. Human cerebral organoids recapitulate gene expression programs of fetal neocortex development. *Proceedings of the National Academy of Sciences of the United States of America*, 112(51):15672–15677, 2015.

[17] Blue B Lake, Rizi Ai, Gwendolyn E Kaeser, Neeraj Salathia, Yun C Yung, Rui Liu, Andre Wildberg, Derek Gao, Holim Fung, Song Chen, et al. Neuronal subtypes and diversity revealed by single-nucleus rna sequencing of the human brain. *Science*, 352(6293):1586–1590, 2016.

[18] Dmitry Usoskin, Alessandro Furlan, Saiful Islam, Hind Abdo, Peter Lonnerberg, Daohua Lou, Jens Hjerlingleffler, Jesper Z Haeggstrom, Olga Kharchenko, Peter V Kharchenko, et al. Unbiased classification of sensory neuron types by large-scale single-cell rna sequencing. *Nature Neuroscience*, 18(1):145–153, 2015.

[19] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah S Andrews, Andrew Yiu, Tamir Chandra, Kedar Nath Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, et al. Sc3: consensus clustering of single-cell rna-seq data. *Nature Methods*, 14(5):483–486, 2017.

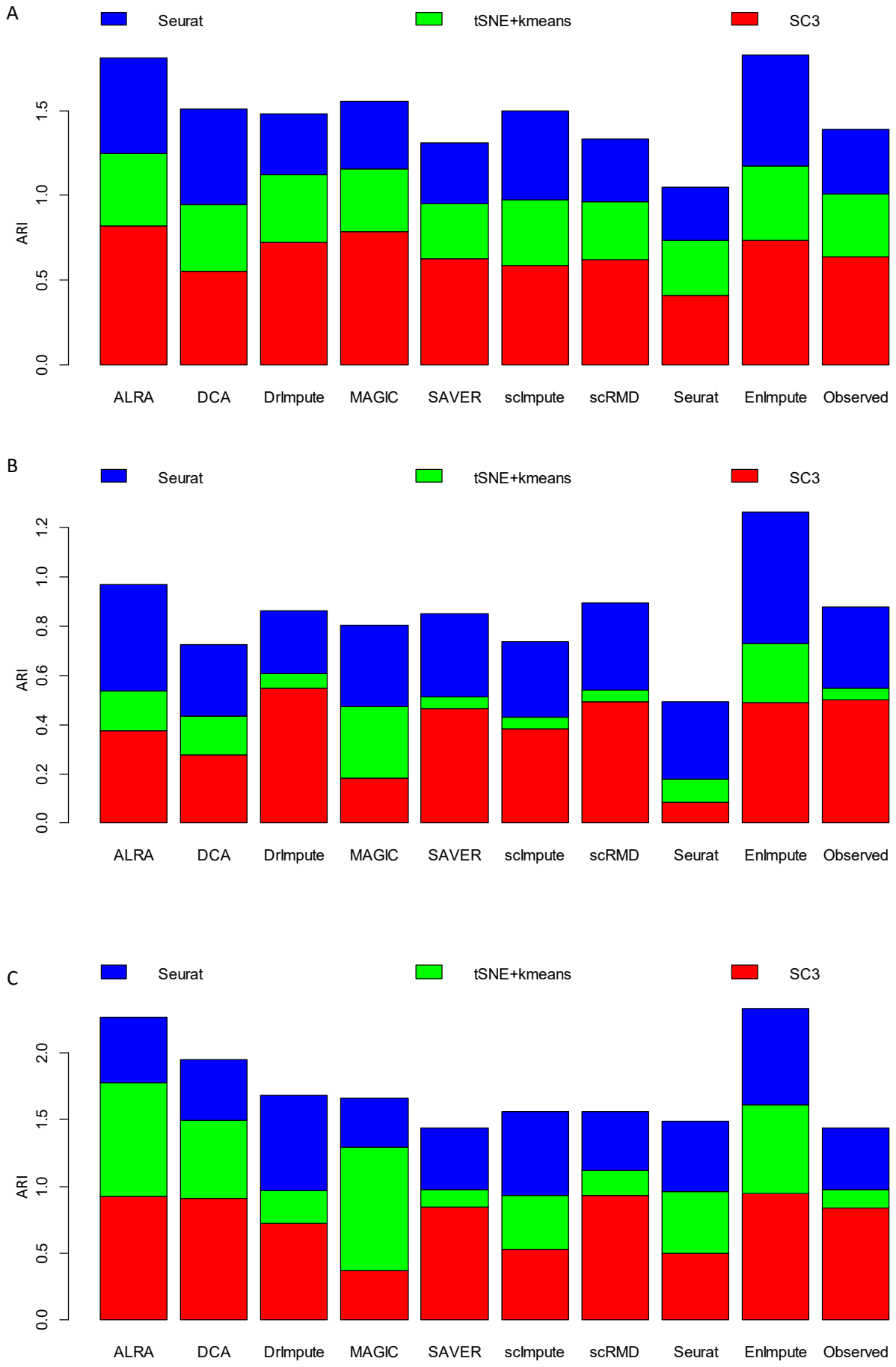

Figure S13: The ARIs of the clustering results of different imputation methods. The Y-axis represents the ARI scores. (A) Camp dataset, (B) Lake dataset, (C) Usoskin dataset.

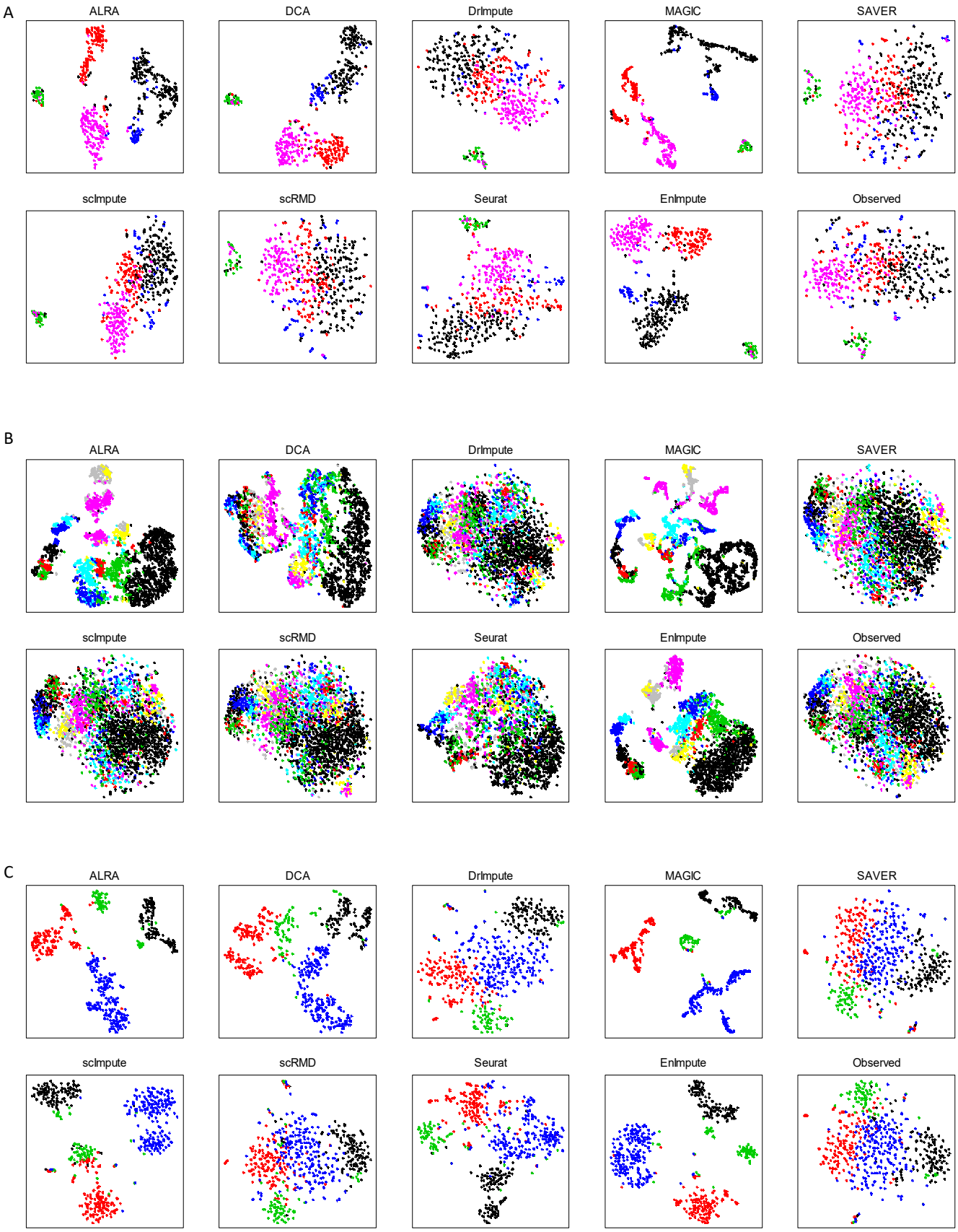

Figure S14: t-SNE visualization of the observed data and the data imputed by different methods. The cells are color-coded by the known cell types. (A) Camp dataset, (B) Lake dataset, (C) Usoskin dataset.

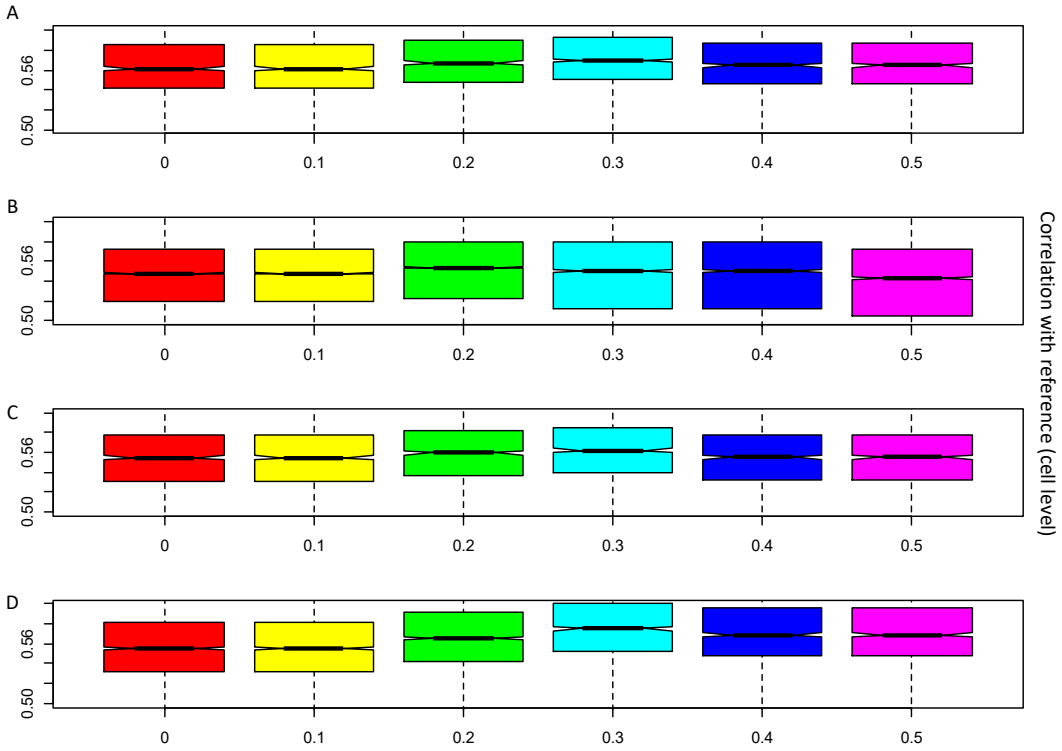

Figure S15: Performance of different values of $trim$ measured by cell-wise correlation with the reference data. For each plot, the X-axis represents the values of $trim$, and the Y-axis represents the Pearson cell-wise correlation across genes between the reference and imputed data. (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

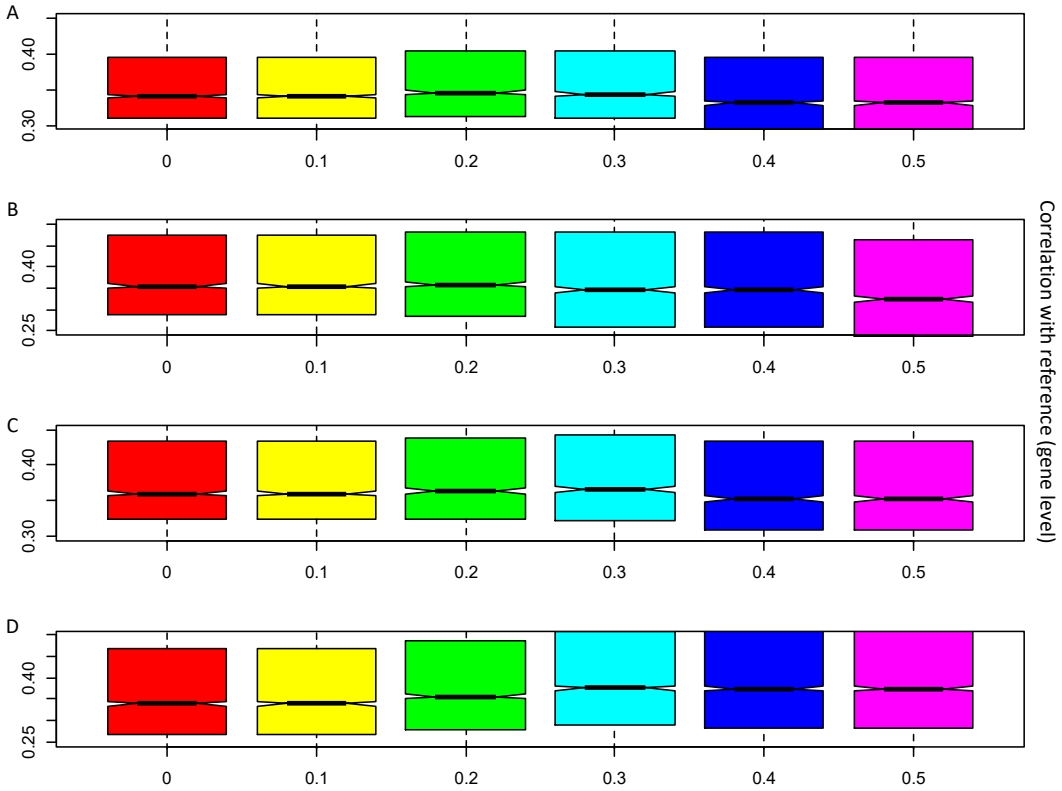

Figure S16: Performance of different values of $trim$ measured by gene-wise correlation with the reference data. For each plot, the X-axis represents the values of $trim$, and the Y-axis represents the Pearson gene-wise correlation across cells between the reference and imputed data. (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

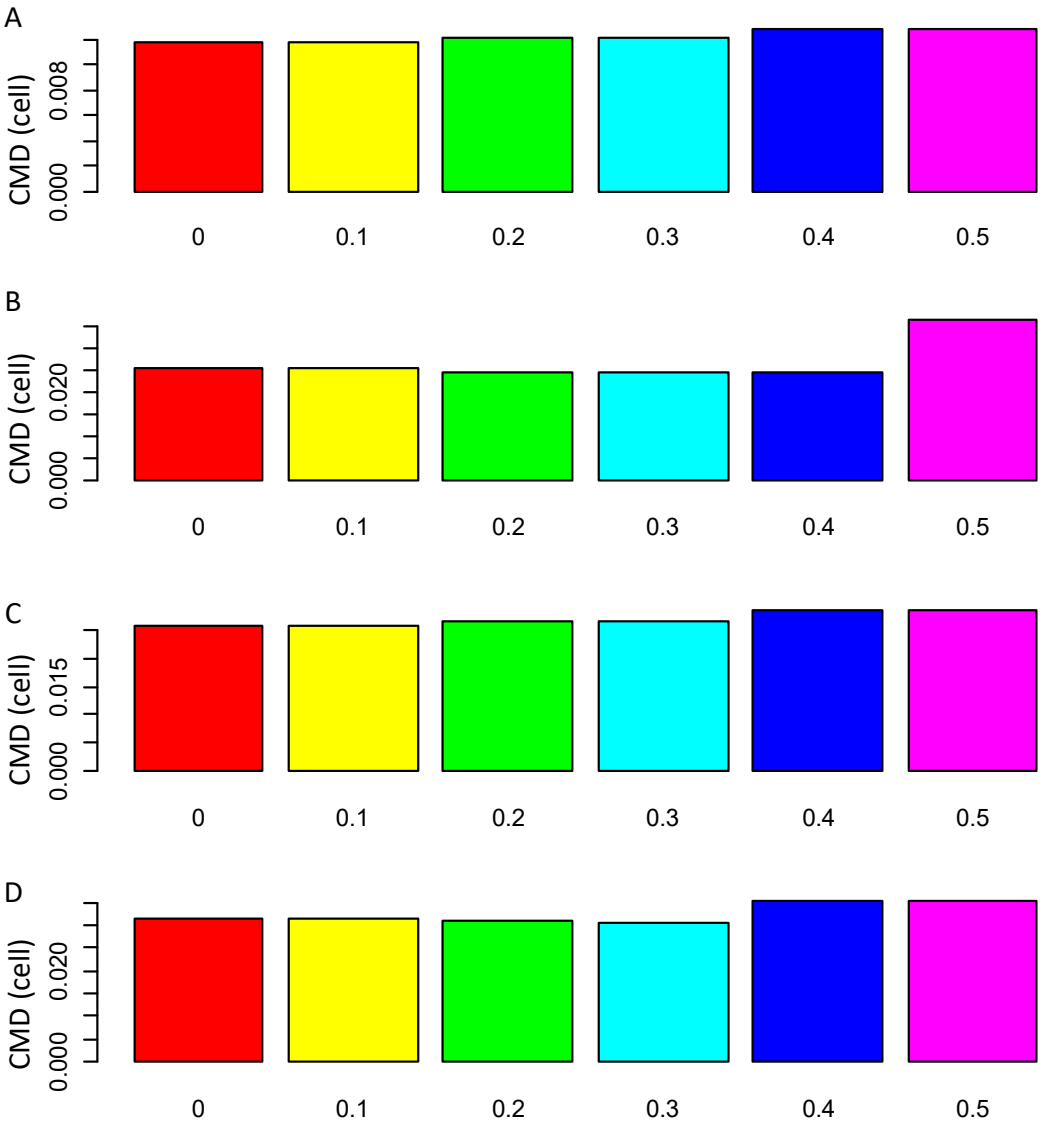

Figure S17: Performance of different values of $trim$ measured by the recovery of cell-to-cell correlation matrices, as measured by correlation matrix distance. For each plot, the X-axis represents the values of $trim$, and the Y-axis represents the correlation matrix distance between the Pearson cell-to-cell correlation matrix derived from the imputed matrix and that derived from the reference matrix. (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.

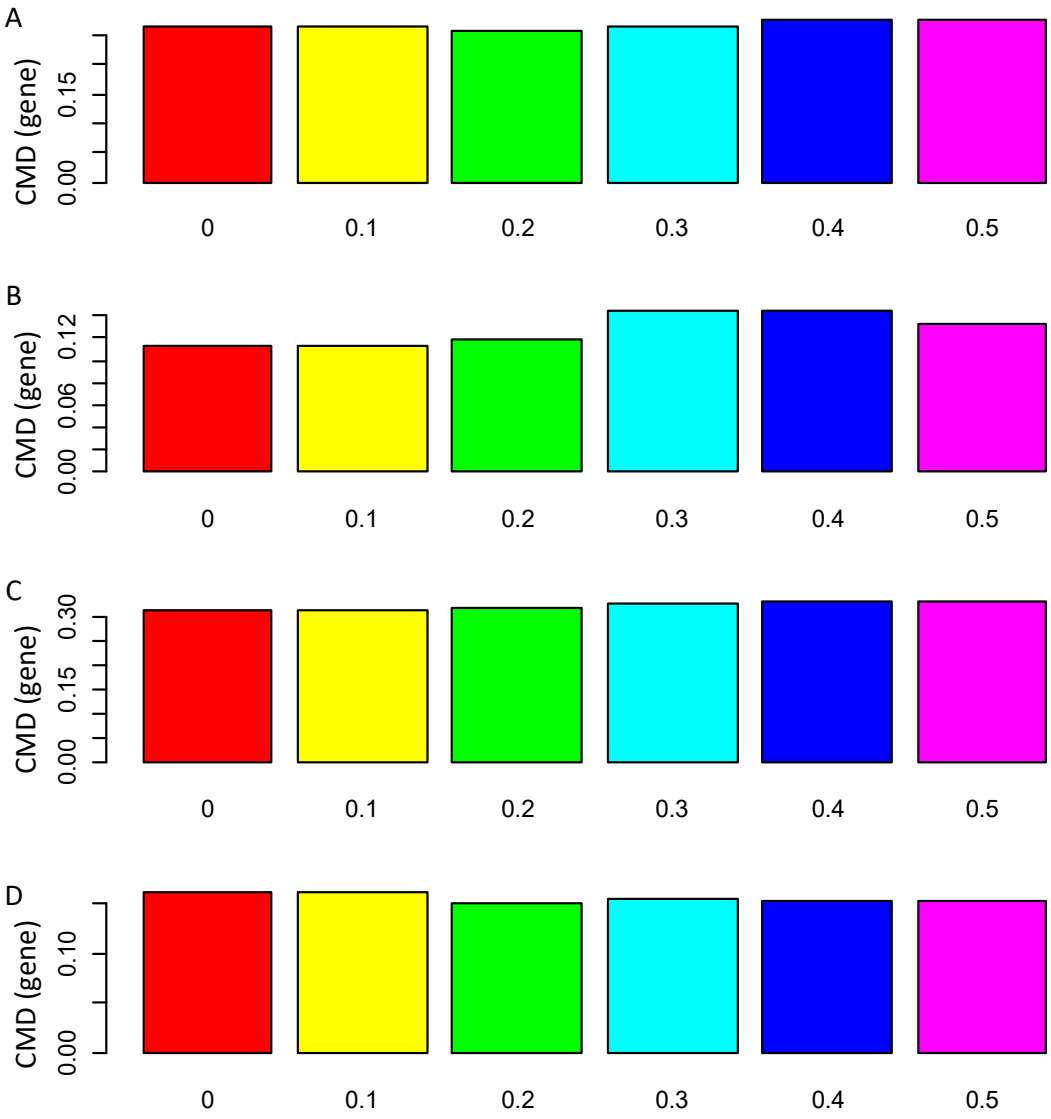

Figure S18: Performance of different values of $trim$ measured by the recovery of gene-to-gene correlation matrices, as measured by correlation matrix distance. For each plot, the X-axis represents the values of $trim$, and the Y-axis represents the correlation matrix distance between the Pearson gene-to-gene correlation matrix derived from the imputed matrix and that derived from the reference matrix. (A) Baron dataset, (B) Chen dataset, (C) Manno dataset, (D) Zeisel dataset.