

博士論文（要約）

**Comprehensive evaluation of preprocessing
methods for visualizing single-cell RNA-seq
count data**

（単一細胞 RNA-seq カウントデータ視覚
化のための前処理法の評価）

張 子龍

Zilong Zhang

Comprehensive evaluation of preprocessing methods for visualizing single-cell RNA-seq count data

Zilong Zhang

A thesis submitted for the degree of

Doctor of Science

To the Department of Biotechnology,

Graduate School of Agricultural and Life Sciences,

The University of Tokyo

Supervisor

Prof. Kentaro Shimizu

Abstract

Single-cell RNA sequencing (scRNA-seq) has enabled researchers to study gene expression in cell level. However, due to extremely low amounts of transcripts in one single cell and the technical loss during the reverse transcription, the visualization of scRNA-seq data is very tricky. To solve this problem, many preprocessing (including imputation and smoothing) methods have been proposed. We collected and compared fifteen popular preprocessing and visualization methods, together with one denoising autoencoder preprocessing model we proposed. By comparing the visualization results of all methods in simulated data and real scRNA-seq data, we got the conclusion that the preprocessing performance for visualizing scRNA-seq count data varies greatly depending on the number of genes. PHATE or DrImpute is practically recommended.

Contents

1 Introduction	4
1.1 Single-cell RNA sequencing technology.....	4
1.2 Preprocessing and dimensionality reduction.....	5
1.3 The purpose of this study.....	6
2 Materials and Methods	7
2.1 Simulated and real datasets	7
2.1.1 Simulated data (Splatter and powsimR)	7
2.1.2 Real datasets.....	8
2.2 Preprocessing and dimensionality reduction.....	11
2.2.1 Neural network (autoencoder)	11
2.2.2 DAE model	12
2.2.3 Preprocessing and visualization methods.....	14
2.3 Evaluation metrics.....	177
2.3.1 Adjusted rand index (ARI).....	18
2.3.2 Normalized mutual information (NMI)	18
2.3.3 Homogeneity score (HOMO)	18
3 Results and Discussion	19
3.1 Splatter simulated data	19
3.2 powsimR simulated data	22
3.3 Real datasets.....	错误!未定义书签。
4 Conclusion.....	19
Acknowledgments	21
References	22
Supporting Information	40

List of figures

Figure 1. Typical procedure for an exploratory analysis of scRNA-seq data.....	4
Figure 2. An example of real scRNA-seq count matrix.	5

Figure 3. Model of a simple neuron.....	10
Figure 4. Model of a simple network.....	11
Figure 5. Model of a simple autoencoder	12
Figure 6. Framework of DAE model.....	13
Figure 7. ARI boxplot of Splatter simulated data.....	错误!未定义书签。
Figure 8. ARI boxplot of powsimR simulated data.....	错误!未定义书签。
Figure 9. ARI boxplot of powsimR simulated data (different sample sizes).....	错误!未定义书签。
Figure 10. ARI boxplot of powsimR simulated data (different gens numbers)..	错误!未定义书签。
Figure 11. ARI boxplot of real data	错误!未定义书签。
Figure 12. ARI boxplot of real data (different species).....	28
Figure 13. ARI boxplot of real data (different sample sizes).....	29
Figure 14. ARI boxplot of real data (different gens numbers).....	30
Figure 15. ARI boxplot of real data (different models).....	31

List of supporting figures

Figure S1. Best latent dimension of DAE model	46
Figure S2. NMI boxplot of Splatter simulated data.....	47
Figure S3. HOMO boxplot of Splatter simulated data.....	48
Figure S4. NMI boxplot of powsimR simulated data	49
Figure S5. HOMO boxplot of powsimR simulated data	50
Figure S6. NMI boxplot of powsimR simulated data (different sample sizes).....	51
Figure S7. HOMO boxplot of powsimR simulated data (different sample sizes)	52
Figure S8. NMI boxplot of powsimR simulated data (different gens numbers).....	53
Figure S9. HOMO boxplot of powsimR simulated data (different gens numbers)	54
Figure S10. NMI boxplot of real data	55
Figure S11. HOMO boxplot of real data	56
Figure S12. NMI boxplot of real data (different species).....	57
Figure S13. HOMO boxplot of real data (different species).....	58
Figure S14. NMI boxplot of real data (different sample sizes)	59
Figure S15. HOMO boxplot of real data (different sample sizes)	60
Figure S16. NMI boxplot of real data (different gens numbers)	61
Figure S17. HOMO boxplot of real data (different gens numbers)	62
Figure S18. NMI boxplot of real data (different models).....	63
Figure S19. HOMO boxplot of real data (different models)	64

List of tables

Table 1. Splatter parameters.....	7
Table 2. powsimR parameters.....	9

Table 3. List of 15 real datasets.....	9
Table 4. Summary of 16 pipelines	17
Table 5. Average ARI values for individual conditions of Splatter simulated data.	25
Table 6. Average ARI values for individual conditions of powsimR simulated data.	26
Table 7. Average ARI values for individual real datasets.	27
Table S1. Average NMI values for individual conditions of Splatter simulated data.	27
Table S2. Average HOMO values for individual conditions of Splatter simulated data.	28
Table S3. Average NMI values for individual conditions of powsimR simulated data.	29
Table S4. Average HOMO values for individual conditions of powsimR simulated data.	30
Table S5. Average NMI values for individual conditions of real datasets.	31
Table S6. Average HOMO values for individual conditions of real datasets.	32

1 Introduction

1.1 Single-cell RNA sequencing technology

The rapid development of high-throughput sequencing (HTS) methods has led to many new discoveries in biological sciences. RNA sequencing (RNA-seq) is a transcriptome profiling approach that uses HTS (Wang et al., 2009). In traditional bulk RNA-seq, extraction of mRNA is done from a population of cells. Therefore, bulk RNA-seq can only measure average expression levels (Shapiro et al., 2013). This means the ability to evaluate gene expression at the cell level is lost, which may be a hindrance as each cell in an organism is unique (Kolodziejczyk et al., 2015). Single-cell RNA sequencing (scRNA-seq) makes it possible for researchers to study gene expression for each single cell (Stegle et al., 2015).

Figure 1 shows the typical procedure for analysis of scRNA-seq; single cells are first dissociated from different tissues and libraries developed from protocols such as the CEL-seq2 (Hashimashony et al., 2016) and Drop-seq (Macosko et al., 2015) are used to dispose them. To generate a count matrix, quantification tools such as Kallisto (Bray et al., 2016) and Salmon (Patro et al., 2017) have been commonly used in conjunction with reference sequences such as RefSeq (O’Leary et al., 2016).

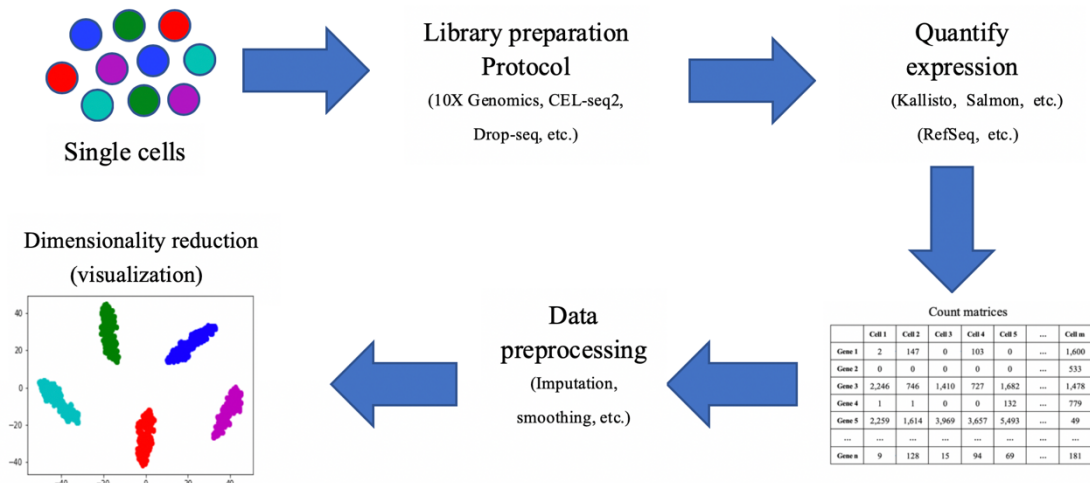


Figure 1. Typical procedure for an exploratory analysis of scRNA-seq data.

This study focuses on the last two steps (i.e., preprocessing and dimensionality reduction).

Although scRNA-seq offers a lot of benefits in cell-level biological studies, there are still many challenges in its practical implementation; Due to extremely low amounts of input RNA obtained from individual cells and the technical loss during the reverse transcription, the sequencing per cell is shallow and the scRNA-seq data are usually very noisy (Haque et al., 2017). The frequent occurrence of dropout events is the most significant feature of scRNA-seq data. Dropout events are defined as

genes whose expression has been wrongly reported as absent during the sequencing step due to the low amount of RNA material or technical errors (Moon et al., 2018).

Data analysis of scRNA-seq data often starts from a raw count matrix that contains tens of thousands of genes or dimensions in rows and hundreds or thousands of cells in columns. As shown in Figure 2, the raw count matrix contains many entries with zero data values, likely due to dropout events. Therefore, an important step when analyzing scRNA-seq count data is preprocessing (including imputation and smoothing/de-noising) to recover the true values of the missing data entries.

	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	...	Cell m
Gene 1	2	147	0	103	0	...	1,600
Gene 2	0	0	0	0	0	...	533
Gene 3	2,246	746	1,410	727	1,682	...	1,478
Gene 4	1	1	0	0	132	...	779
Gene 5	2,259	1,614	3,969	3,657	5,493	...	49
...
Gene n	9	128	15	94	69	...	181

Figure 2. An example of real scRNA-seq count matrix.

A part of the real count data of Deng et al. (2014) is shown. Since this matrix consists of 22,431 genes \times 268 cells, n is 22,431 and m is 268. This data contains a total of 3,624,354 zero counts (i.e., %Dropouts = 60.3%).

1.2 Preprocessing and dimensionality reduction

Since the scRNA-seq count matrix is high dimensional, better visualization and analysis are achieved through dimensionality reduction. Classical dimensionality reduction approaches include the linear principal component analysis (PCA) and the t-distributed stochastic neighbor embedding (t-SNE) (Van der Maaten & Hinton, 2008). PCA is a linear approach that generates reduced dimensions by maximizing the captured residual variance in each further dimension, while t-SNE is a non-linear approach whose dimensions focus on capturing local similarity at the expense of global structure (Luecken & Theis, 2019). However, their direct use on raw scRNA-seq count data is insufficient because they do not capture the characteristics of scRNA-seq data with high dropout rates. In this study, the two methods (PCA and t-SNE) specialized in dimensionality reduction and visualization were defined as the first category.

In recent years, several methods that perform both preprocessing and visualization have been developed. These methods include the Zero-Inflated Factor Analysis (ZIFA) (Person & Yau, 2015), the Potential of Heat-diffusion for Affinity-based Trajectory Embedding (PHATE) (Moon et al., 2018), Clustering through Imputation and Dimensionality Reduction (CIDR) (Lin et al., 2018) and the Markov Affinity-based Graph Imputation of Cells (MAGIC) (van Dijk et al., 2018). These methods have so far yielded many new discoveries from scRNA-seq data. The four methods were defined as the second category.

The third category is defined as methods mainly dedicated for preprocessing (i.e., imputation and smoothing). This study includes a total of ten methods: Single-cell Analysis Via Expression Recovery (SAVER) (Huang et al., 2018), SAVER-X (Wang et al., 2019), scImpute (Li & Li, 2018), Deep Count Autoencoder network (DCA) (Eraslan et al., 2019), Autoencoder based imputation (autoImpute) (Talwar et al., 2018), DrImpute (Gong et al., 2018), LSimpute (Bø et al., 2004), k-Nearest Neighbor Smoothing (kNN-smoothing) (Wagner et al., 2018), scRMD (Chen et al., 2018), and a new method proposed here.

Of these, DCA denoises scRNA-seq data by learning the underlying true zero-noise data manifold using an autoencoder framework. An autoencoder is a type of artificial neural network trying to learn a representation of high dimensional data in an unsupervised manner. Several recent studies employed autoencoders in molecular biology. Ding et al. (2018) proved that autoencoder could be used to preserve global structure of high-dimensional scRNA-seq data. Wang & Gu (2017) elaborated autoencoder could help make clearer separation of rare cell types. Way & Greene (2018) have shown that the ability for an autoencoder to learn a manifold with meaningful relationships between samples.

1.3 The purpose of this study

A visual representation of the scRNA-seq count data with two or three dimensions (2D or 3D) has led to the discovery of new cell types and/or subtypes. This study mainly focuses on preprocessing methods for visualizing scRNA-seq data. Since different researchers are continually reporting new preprocessing and visualization methods, it has become difficult to determine which method is suitable for one's own data. The purpose of this study is to evaluate ten preprocessing methods with t-SNE as well as four other methods. In this framework, this study also aims to use an autoencoder-based preprocessing method (called DAE).

Comprehensive analysis with both simulated and real datasets provides sound recommendations regarding an optimal guideline for the dataset at hand. The rest contents of this thesis are as follows: Section 2 describes the simulated and real scRNA-seq datasets, methods (including the DAE model), and evaluation metrics. Section 3

evaluates the performances of a total of 16 analysis pipelines with various kinds of datasets. Section 4 concludes the whole thesis.

2 Materials and Methods

2.1 Simulated and real datasets

2.1.1 Simulated data (Splatter and powsimR)

Real data has problems such as doublet (i.e., a droplet has captured at least two cells), and the truth is not always clear. Therefore, simulation analysis is important. There are several ways to generate simulation data. Most of the method comparison papers used so far are Splatters (Zappia et al., 2017), but there are other good ones like powsimR (Vieth et al., 2017). I used both programs in this thesis. A total of 17 conditions were evaluated: eight from Splatter (version 1.8.0) and nine from powsimR (version 1.1.4).

Each Splatter dataset contained 1,000 cells split into 2–5 groups (or cell types) and 1,000–5,000 genes of which 5–20% were differentially expressed (DE) across the groups (Table 1). The `splatSimulate()` R function with following options was used to generate Splatter datasets: “method” for indicating the simulation method, “batchCells” for the number of cells, “group.prob” for the probability that a cell comes from a group, “nGenes” for the number of genes, “de.prob” for the probability that a gene is DE in a group, and “dropout.mid” for the midpoint parameter used in the dropout logistic function. (The midpoint and shape parameters for the dropout function are estimated by fitting a logistic function to the relationship between the log means of the normalized counts and the proportion of samples that are zero for each gene).

For example, the “splat_1” simulated data was produced using the following parameters: `method = “groups”, batchCells = 1000, group.prob = c(0.5, 0.5), nGene = 1000, de.prob = 0.2, and dropout.mid = 0.8`. As another example, the “splat_5” simulated data was produced using the following parameters: `method = “groups”, batchCells = 1000, group.prob = c(0.5, 0.5), nGene = 5000, de.prob = 0.05, and dropout.mid = 0.8`.

Table 1. Splatter parameters. The “%Dropouts” indicates percentage of zero counts in the count matrix.

Names	Cells	Groups	Genes	%DE	%Dropouts
splat_1	1,000	2	1,000	20%	14.6%
splat_2	1,000	5	1,000	20%	14.5%
splat_3	1,000	2	5,000	20%	33.3%
splat_4	1,000	5	5,000	20%	33.3%
splat_5	1,000	2	5,000	5%	33.0%
splat_6	1,000	5	5,000	5%	33.0%

splat_7	1,000	2	5,000	20%	33.3%
splat_8	1,000	5	5,000	20%	33.3%

Each powsimR dataset contained 200–2000 cells split into three groups and 10,000–30,000 genes of which 10% DE (Table 2). The simulateCounts() R function with following options was used to generate powsimR datasets: “n” for the number of cells that comes from a group, “ngenes” for the number of genes, “p.DE” for the probability that a gene is DE in a group. For example, the “powsim_1” simulated data was produced using the following parameters: n = c(100, 50, 50), ngenes = 10000, and p.DE = 0.1. Different from the simulation settings in Splatter, the proportions of cells in individual groups differ. The way to specify with the n option is equivalent to group.prob = c (0.5, 0.25, 25) in Splatter.

Table 2. powsimR parameters.

Names	Cells	Groups	Genes	%DE	%Dropouts
pow_1	200	3	10,000	10%	65.3%
pow_2	1,000	3	10,000	10%	65.1%
pow_3	2,000	3	10,000	10%	65.0%
pow_4	200	3	20,000	10%	65.9%
pow_5	1,000	3	20,000	10%	66.0%
pow_6	2,000	3	20,000	10%	65.2%
pow_7	200	3	30,000	10%	65.5%
pow_8	1,000	3	30,000	10%	65.6%
pow_9	2,000	3	30,000	10%	65.8%

2.1.2 Real datasets

For real data, 15 raw count datasets (five from humans and ten from mice) were obtained from original papers. Each real dataset contained 56–3,005 cells split into 3–32 groups and 19,020–41,480 genes. The details were shown in Table 3.

Table 3. List of 15 real datasets. “Names” consists of the “*first author’s name_species*”. For example, the name “Biase_M” indicates that the first author’s name is “Biase” and the species of this dataset is “Mouse”.

Names	Cells	Groups	Genes	Pubmed	%Dropouts
Camp_H	777	7	19,020	28614297	68.1%
Wang_H	635	8	19,950	27364731	71.6%
Zeisel_H	3,005	9	19,972	25700174	81.2%
Darmanis_H	90	9	20,214	26060301	80.8%
Yan_H	90	6	20,214	23934149	45.1%
Deng_M	268	6	22,431	24408435	60.3%
Manno_M	2,150	32	23,530	27716510	87.3%
Klein_M	2,717	4	24,175	26000487	65.8%
Romanov_M	2,881	7	24,341	27991900	87.7%
Usoskin_M	622	4	25,334	25420068	84.5%
Biase_M	56	4	25,737	25096407	50.9%
Fan_M	69	6	26,357	26201400	57.9%
Kolodziejczyk_M	704	3	38,653	26000846	70.7%
Xin_H	1,600	8	39,851	27667665	85.6%
Goolam_M	124	5	41,480	27015307	67.9%

2.2 Preprocessing and dimensionality reduction

2.2.1 Neural network (autoencoder)

Artificial neural networks are computing systems that are inspired by biological neural networks, which is usually often just called neural networks for short. The basic units for neural networks are artificial neurons. One neuron is a simple computational unit, which uses an activation function to produce an output from weighted input. Figure 3 shows a simple neuron model.

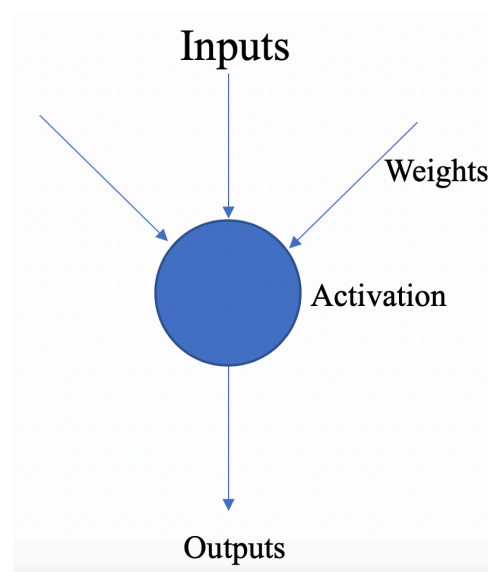


Figure 3. Model of a simple neuron.

Similar to linear regression, each weighted input also has a bias. The inputs for a neuron are summed and then passed to an activation function, which is a function applied at the end of the computational process of each neuron. Once I choose a nonlinear activation function, the network could combine the inputs in complex ways, which makes the network could deal with complex or noisy data. Currently most widely used activation functions include (1) Rectified Linear Units (ReLU), (2) sigmoid, and (3) tanh.

(1) ReLU is the most widely used activation function which is defined as:

$$ReLU(x) = \max \{0, x\}$$

x stands for the input of a neuron. It creates a nonlinear function by flattening all negative values to zero. It is simple but critical for outputting only nonnegative values.

(2) sigmoid is primarily used in classification. It ranges from 0 to 1. The sigmoid function is defined as:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

The output of a sigmoid function is closer to one, the more certain we can be for a prediction.

(3) tanh is an activation function also widely used for classification. The tanh function is defined as:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

It ranges from -1 to 1 and is usually used for binary classification tasks.

Several neurons are put together to form a neuron network. Each row of neurons is called a layer. A neuron network is often composed with input layer, hidden layer and output layer. The input layer takes input from the dataset. Layers after the input layer are called hidden layers. If there are many hidden layers in the networks, the neural networks are often called deep learning. The final layer of network is called output layer, this layer usually gives us the final prediction results. Figure 4 shows a simple network.

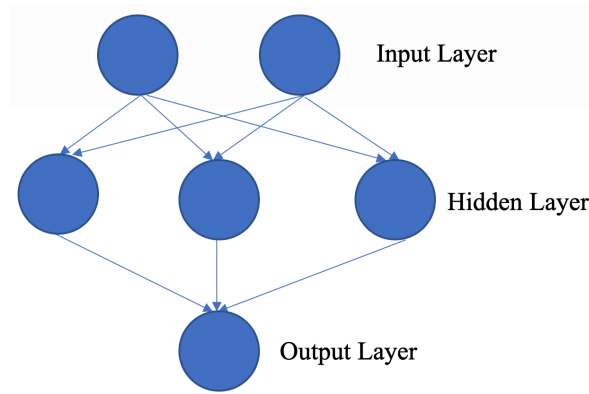


Figure 4. Model of a simple network.

After building the architecture of neural networks, the next step is to update the weights. Optimizers are the algorithms to minimize the loss function. The classical optimizer is called stochastic gradient descent. The network does the forward pass by using one row of data at a time until gets to final output value. The output value is compared to expected output and calculate an error. The error is propagated back through the network to update the weights.

There are several items need to be explained for neural network. **Batch size** means the number of samples that will be propagated through the network. **Learning rate** is the amount that the weights are updated during training, higher learning rate will update the weights faster. All the training data were used to repeat training process and in order to get better trained neural networks, training data is often iteratively used several times. One **epoch** means one round of updating the network using the training data. **Optimizer** is used to shape and mold your model by updating the model in response to the output of the loss function.

After the training step, we could make predictions on the test data. A neural network model could be saved by just saving the topology and the weights. When you need to make a prediction, you can provide the input to the neural network and perform a forward-pass to get the prediction results.

In an autoencoder network, the output layer had the same number of nodes as the input layer. Instead of predicting, the output layer intends to reconstruct the input layer by learning the representation of the input data in the hidden layer. To achieve the goal of learning an efficient representation, autoencoder first maps the original data to a lower dimensional space. Then try to map the lower dimensional representation back to the original space to make them as close as possible. The representing data could be used as a dimensionality reduction of the input data which ignores noise in input data (Aleksandar, A., 2008). Figure 5 shows a simple autoencoder network.

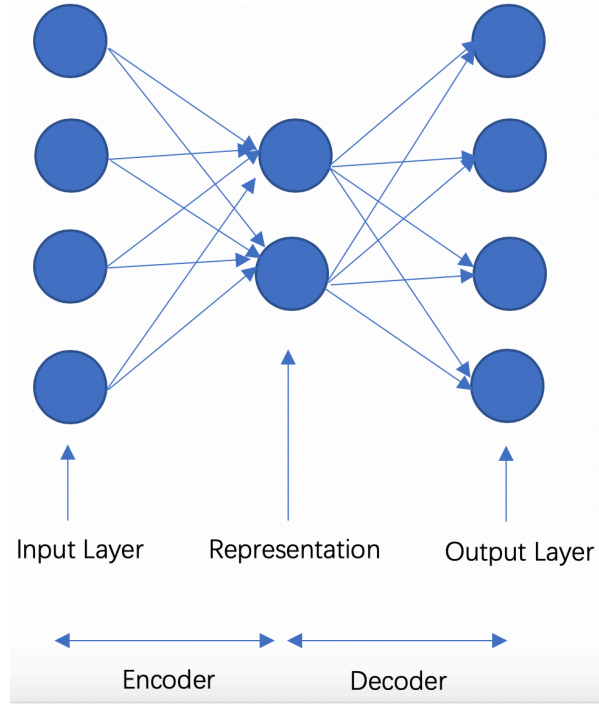


Figure 5. Model of a simple autoencoder network.

Denoising Autoencoder is an extension of traditional autoencoders especially useful for analyzing noisy data. It first partially corrupts the original data, then using the corrupted data to train the network to recover the original data. The underlying assumption is that to recover from corrupted data to original data, the lower dimensional representation needs to extract true useful features of the original data.

2.2.2 DAE model

To get better visualization results of scRNA-seq data, I developed a data preprocessing method DAE. DAE first randomly corrupts input, and by training to reconstruct the raw input data without noise. Due to the noisy characteristic of scRNA-seq data, DAE would be a good choice as a feature extraction method to get the low dimension representation of raw data. I use the DAE network to remove noisy signals in scRNA-seq data and get the lower dimension representation of the raw data. Then the t-SNE is applied to the lower representation data to get the two-dimensional visualization. Figure 6 shows the framework of DAE model.

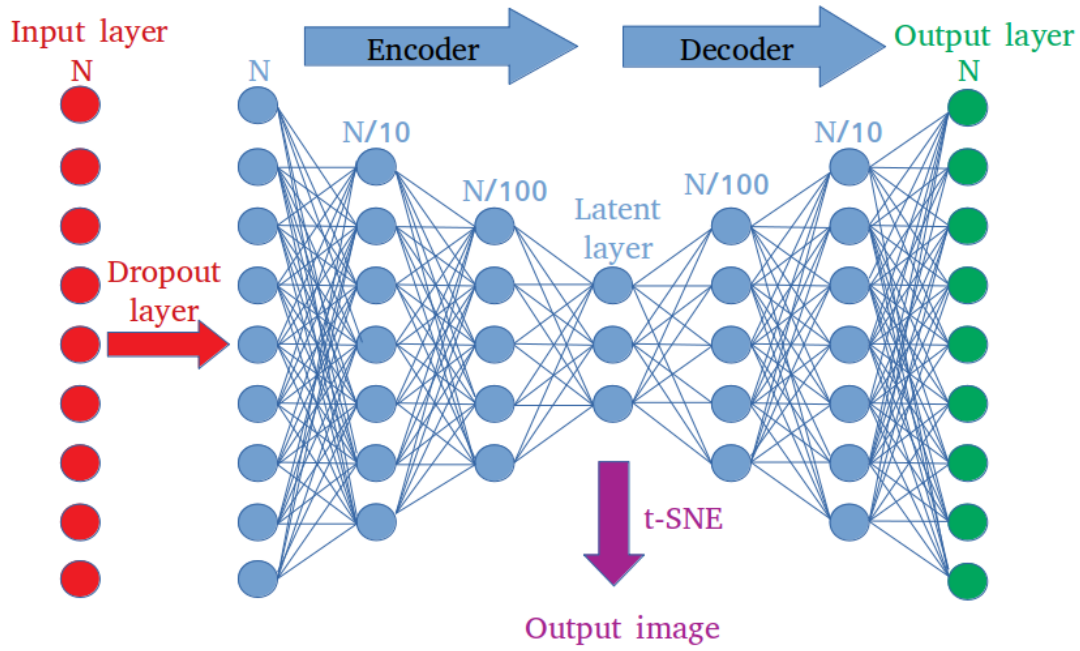


Figure 6. Framework of DAE model.

The model is divided and explained by (1) Input layer and Output layer, (2) Dropout layer, (3) Encoder network, (4) Latent layer, (5) Decoder network, and (6) Hyperparameters.

(1) Input layer and Output layer

The input of DAE model was raw gene expression read counts matrix from scRNA-seq data. The number of nodes (N) in the input layer was set same as the gene numbers in the input data. As the output layer was designed to reconstruct the input layer, the number of nodes in output layer was set just the same as the input layer.

(2) Dropout layer

A dropout layer was designed to randomly change a proportion of matrix items into zeros. The dropout layer made our model more suitable for analyzing scRNA-seq data. To recover from corrupted data to original data, the lower dimensional representation needs to extract useful features of the original data. The default dropout rate was set to 0.5.

(3) Encoder network

The encoder network was designed as a fully connected neural network. The number of nodes in the first hidden layer was set to 1/10 of the number of genes in the input layer. The number of nodes in the second hidden layer was set to 1/10 of the number of nodes in the first hidden layer.

(4) Latent layer

In this layer, I tried the number of nodes to five values (10, 20, 30, 40 and 50) and evaluated the performance of these five structures. To determine the best node numbers for the latent layer, five possible numbers (10, 20, 30, 40 and 50) were searched. With 20 trials using the “splat_1” simulated data (Table 1), the DAE model with 30 latent dimensions shown the best performance overall (Figure S1). Therefore, the output of 30 latent dimensions was used as input for t-SNE when performing the DAE pipeline.

(5) Decoder network

The decoder network was used to recover the original expression matrix. The decoder network structure was designed symmetric to the structure of encoder layer to reconstruct the input data.

(6) Hyperparameters

In general, methods have substantial sensitivity to hyperparameters (Hu and Greene, 2019). In this study, Adam (Kingma & Ba, 2014) was used as the optimizer with the loss function of mean square error. The ReLU activation function was used for all layers. The learning rate was set to 0.0001 and epoch was set to 50. To alleviate the overfitting, these tuning was performed using two Splatter simulated datasets (“splat_1” and “splat_2”) and two real datasets (“Darmanis_H” and “Pollen_H” (Pollen et al., 2014)). These four datasets correspond to training set. Tuning in the latent layer described above is also a type of hyperparameter tuning.

2.2.3 Preprocessing and visualization methods

A total of 16 pipelines with raw count data as input and output reduced data were evaluated. As described in the section 1.2, they can be divided into three categories. The first category consists of two pipelines: (1) PCA and (2) t-SNE. Since they only perform dimensionality reduction (i.e., visualization). Therefore, only dimensionality reduction into 2D was performed based on the input data.

The second category consist of four pipelines: (3) ZIFA, (4) PHATE, (5) CIDR, and (6) MAGIC. They perform from preprocessing to visualization in one way. As for

methods in the second category, the preprocessing work was done for each method following its instruction then followed by t-SNE visualization.

The third category consist of ten preprocessing methods: (7) SAVER, (8) SAVER-X, (9) scImpute, (10) DCA, (11) autoImpute, (12) DrImpute, (13) LSimpute, (14) kNN-smoothing, (15) scRMD, and (16) DAE. Since they don't have a means of visualization by themselves, t-SNE was subsequently applied to the outputs of these preprocessing methods for obtaining the final 2D representation. With descriptions for individual pipelines, the summary is also shown in Table 4.

First category (two pipelines)

(1) PCA (sklearn ver. 0.20.3) (https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/decomposition/_pca.py)

The “`decomposition.PCA(n_components=2).fit_transform()`” python function was used to perform visualization.

(2) t-SNE (sklearn ver. 0.20.3) (https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/manifold/tests/test_t_sne.py)

The “`manifold.TSNE(n_components=2).fit_transform()`” python function was used to perform visualization.

Second category (four pipelines)

(3) ZIFA ver. 0.1 (<https://github.com/epierson9/ZIFA>)

This method uses a zero-inflated negative binomial (ZINB) model that fits scRNA-seq data well. The `block_ZIFA.fitModel()` python function was used to perform visualization.

(4) PHATE ver. 0.4.1 (<https://github.com/KrishnaswamyLab/PHATE>)

This method captures both local and global nonlinear structure in data by an information-geometric distance between data points. The `phate.PHATE().fit_transform()` python function was used to perform visualization.

(5) CIDR ver. 0.1.5 (<https://github.com/VCCRI/CIDR>)

This is an ultrafast algorithm that uses a novel yet very simple implicit imputation approach to alleviate the impact of dropouts in scRNA-seq data in a principled manner. For example, the following R function is used to perform visualization for “dat1” data:

```
scdat1 <- scDataConstructor(as.matrix(dat1)), scdat1 <-
determineDropoutCandidates(scdat1), scdat1 <- wThreshold(scdat1), scdat1 <-
scDissim(scdat1), scdat1 <- scPCA(scdat1), scdat1 <- nPC(scdat1), nCluster(scdat1),
scdat1 <- scCluster(scdat1), data_cidr <- scdat1@PC[,c(1,2)].
```

(6) MAGIC ver. 1.5.3 (<https://github.com/KrishnaswamyLab/MAGIC>)

This method uses a ZIMB model. MAGIC employs information across similar cells, via data diffusion, to denoise the cell count matrix and fill in missing transcripts. The `apply_tSNE(magic.MAGIC().fit_transform(Z, genes='all_genes'))` python function was used to perform visualization.

Third category (ten pipelines)

(7) SAVER ver. 1.1.1 (<https://github.com/mohuangx/SAVER>)

This method uses a ZIMB model. This is an expression recovery method for unique molecule index (UMI)-based scRNA-seq data that borrows information across genes and cells to provide accurate expression estimates for all genes.

(8) SAVER-X ver. 0.0.0.9 (<https://github.com/jingshuw/SAVERX>)

This method uses a ZIMB model. SAVER-X employs a deep autoencoder with a Bayesian model. It extracts transferable gene-gene relationships across data from different labs, varying conditions, and divergent species to denoise target new datasets.

(9) scImpute ver. 0.0.9 (<https://github.com/Vivianstats/scImpute>)

This method uses a ZIMB model. This method automatically identifies likely dropouts, and only perform imputation on these values without introducing new biases to the rest data.

(10) DCA ver. 0.2.3 (<https://github.com/theislab/dca>)

This method uses a ZIMB model. DCA takes the count distribution, overdispersion and sparsity of the data into account using a negative binomial noise model with or without zero-inflation, and nonlinear gene-gene dependencies are captured.

(11) autoImpute ver. 0.11.6 (<https://github.com/divyanshu-talwar/AutoImpute>)

This method learns the inherent distribution of the input scRNA-seq data and imputes the missing values accordingly with minimal modification to the biologically silent genes.

(12) DrImpute ver. 1.0 (<https://github.com/gongx030/DrImpute>)

This method first identifies similar cells based on clustering, and imputation is performed by averaging the expression values from similar cells.

(13) LSimpute (<http://cnv1.engr.uconn.edu:3838/LSImpute>)

This method is designed for microarray experiments generate datasets, which are specialized methods utilizing the least squares principle to estimate missing values using correlations between genes and between arrays.

(14) kNN-smoothing ver. 2.1 (<https://github.com/yanailab/knn-smoothing>)

This is designed to reduce noise by aggregating information from similar cells (neighbors) in a computationally efficient and statistically tractable manner.

(15) scRMD ver. 0.99 (<https://github.com/ChongC1990/scRMD>)

This method models the dropout imputation problem as robust matrix decomposition.

(16) DAE (our method)

The details were described in 2.2.2.

Table 4. Summary of 16 pipelines

Preprocessing	Visualization	Use ZINB model?	Imputation or smoothing
-	PCA	No	-
-	t-SNE	No	-
ZIFA		Yes	-
PHATE		No	-
CIDR		No	-
MAGIC		Yes	-
SAVER	t-SNE	Yes	imputation
SAVER-X	t-SNE	Yes	smoothing
scImpute	t-SNE	Yes	imputation
DCA	t-SNE	Yes	smoothing
autoImpute	t-SNE	No	imputation
DrImpute	t-SNE	No	imputation
LSimpute	t-SNE	No	imputation
kNN-smoothing	t-SNE	No	smoothing
scRMD	t-SNE	No	imputation
DAE (our model)	t-SNE	No	smoothing

2.3 Evaluation metrics

To measure the quality of low-dimensional representation, k -means clustering was applied to the 2D representations of all the pipelines. The clustering results were compared with known groups (cell types). The number of clusters, k , was set to the number of known groups. Three evaluation metrics were used to assess the clustering result: the adjusted rand index (ARI; Hubert & Arabie, 1985), the normalized mutual information (NMI; Kvalseth, TO, 1987), and the homogeneity score (HOMO; Vinh et al., 2010). They have been used to calculate similarity between the clustering results and the known groups (e.g., Wang & Gu, 2018; Hu and Greene, 2019).

2.3.1 Adjusted rand index (ARI)

ARI penalizes both false positive and false negative decisions, where a positive decision means that two cells are clustered into one cluster, while a negative decision means that two cells are clustered into different clusters.

Suppose U is the known cell types, and V denotes the predicted clustering results (the same below). Given a set of n samples, a_i is the number of samples appearing in the i -th cluster of V , b_j is the number of samples appearing in the j -th types of U , and n_{ij} denotes the number of overlaps between the i -th cluster of V and the j -th type of U . The ARI can then be calculated as:

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

Where $()$ denotes a binomial coefficient. A larger ARI value means a higher agreement between two clusters. The maximum ARI value is 1, and the minimal value is 0 in the case of random clusters.

2.3.2 Normalized mutual information (NMI)

The NMI score is calculated as:

$$NMI(U, V) = \frac{2I(U, V)}{H(U) + H(V)}$$

Where $H(U)$ and $H(V)$ denotes entropy of U and V , respectively (the same below). $I(U, V)$ denotes mutual information. NMI score ranges from 0 to 1, 0 stands for no mutual information while 1 stands for perfect correlation.

2.3.3 Homogeneity score (HOMO)

The HOMO expects every cluster to only contain samples from one cell type. Suppose $H(U|V)$ is the cross-entropy of cell types given the cluster V . The Homogeneity score is computed as:

$$HOMO = 1 - \frac{H(U|V)}{H(U)}$$

HOMO is used to represent the level that a cluster only contain samples belongs to a single class. It's bounded between 0 and 1, 0 means low homogeneity and 1 means high homogeneity.

3 Results and Discussion

This chapter is scheduled to published in the form of a scientific journal within 5 years.

本章の内容は、学術雑誌論文として出版する計画があるため公表できない。
5年以内に出版予定。

4 Conclusion

ScRNA-seq provides RNA expressions of a number of individual cells with high resolution. A major challenge when analyzing scRNA-seq count data is on preprocessing (imputation and dimensionality reduction). Therefore, this study focused on this step especially for visualization purpose. The main significance of this study is that many modern methods have been evaluated on different data sets. Although this study has proposed an autoencoder-based approach (i.e., DAE), it is not as important in itself. As recently reported by Hu and Greene (2019), autoencoders have many parameters, and their performance depends heavily on tunable parameters. In this study, hyperparameter tuning for DAE was performed using a small subset of the dataset used. However, the other methods were analyzed with default settings without parameter tuning. Most methods proposed so far have been evaluated by such a means of comparison. The reason for this is probably that a proposed method has to show better performance compared to other existing methods. In this sense, an evaluation of 15 pipelines, excluding DAE, would be justified.

The main contribution of this study is that the preprocessing performance for visualizing scRNA-seq count data varies greatly depending on the number of genes. The conclusion itself is not new because such dependency is also shown in a recent study (Hu and Greene, 2019). However, they only present the results of evaluating five methods (Tybalt, t-SNE, UMAP, ZIFA, and PCA) in a search range of 20,000 to 60,000 genes in the Splatter simulated data. Also, they do not include relatively new methods recommended in this study (i.e., DrImpute and PHATE). In this study, the conclusions obtained only from the analysis of Splatter simulated data in the range of 1,000 to 5,000 genes were that three methods (kNN-smoothing, SAVER, and SAVER-X) could be recommended. This is partially consistent with a recent conclusion derived from the analysis of Splatter simulated data in the range of 1,000 to 5,000 genes where SAVER is better (Andrews and Hemberg, 2019).

However, the current conclusions obtained from the analysis of both powsimR simulated data in the range of 10,000 to 30,000 genes and 15 real datasets in the range of 19,020 to 41,480 genes were that two methods (DrImpute and PHATE) can be practically recommended. As shown in Table 1, Splatter simulated data for genes less than 5,000 does not properly reflect the dropout rate seen in real data, so the recent recommendation reported by Andrew and Hemberg (2018) is somewhat misleading. Of course, this study also has some weaknesses. First, this study did not analyze simulations consisting of tens of thousands of genes in Splatter, which is still widely used in many evaluation studies. Therefore, it is necessary to reinforce the conclusions of this study by performing Splatter simulation analysis consisting of tens of thousands of genes. Second, the parameters of the number of groups were not so varied in this study. This implies that the main conclusion that the difference in the number of genes is important may simply have focused on it. Therefore, it is necessary to perform simulation analysis using parameters of the number of groups close to real data.

Finally, I think the reason why the performance of DAE was not as high as expected was that DAE parameter tuning was mainly performed on Splatter simulated data consisting of a very small number of genes (~1,000 genes). If performance really depends on the number of genes, I think that it is possible to take advantage of the characteristics of DAE with many tunable parameters. For example, high performance could be expected by preparing several DAE models in which parameters have been tuned in advance for several gene sets such as 10000, 15000, and 20000, and using a model having a number of genes close to that of the input data. This kind of research also seems very exciting.

Acknowledgments

I would like to express my great appreciation to Professor Kentaro Shimizu and Associate Professor Koji Kadota for their valuable and constructive suggestions during this research work.

I would also like to extend my thanks to Cui san for generously sharing her time and knowledge to help me understand some important concepts in bioinformatics. I am grateful to all the members of the Bioinformatics and Engineering Lab.

Finally, I wish to thank my family for their support and encouragement throughout my study.

References

- Aleksandar, A. Application of Autoencoders on Single-cell Data (master's thesis). 2008; Retrieved from http://www.dmi.uns.ac.rs/site/dmi/download/master/primenjena_matematika/AleksandarArmacki.pdf.
- Andrews TS & Hemberg M. False signals induced by single-cell imputation. Version 2 F1000Research 2018 Nov 2.
- Becht E, McInnes L, Healy J, Dutertre CA, Kwok IWH, Ng LG, et al. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat Biotechnol.* 2019;37:38-44.
- Biase FH, Cao X & Zhong S. Cell fate inclination within 2-cell and 4-cell mouse embryos revealed by single-cell RNA sequencing. *Genome Res.*, 2014, 24:1787-1796.
- Bø TH, Dysvik B & Jonassen I. LSImpute: accurate estimation of missing values in microarray data with least squares methods. *Nucleic Acids Res.*, 2004, 32(3):e34.
- Bray NL, Pimentel H, Melsted P & Pachter L. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnol.*, 2016, 34:525-527.
- Camp JG, Sekine K & Gerber T, Loeffler-Wirth H. Multilineage communication regulates human liver bud development. *Nature*, 2017, 546:533-538.
- Chen C, Wu C, Wu L, Wang Y, Deng M & Xi R. scRMD: Imputation for single cell RNA-seq data via robust matrix decomposition. *BioRxiv* [Preprint]. 2018; bioRxiv 459404 [posted 2018 Nov 9; cited 2018 Nov 9]. Available from: <https://www.biorxiv.org/content/10.1101/459404v2> doi: 10.1101/459404.
- Darmanis S, Sloan SA, Zhang Y, Enge M, Caneda C, Shuer LM, et al. A survey of human brain transcriptome diversity at the single cell level. *Proc Natl Acad Sci USA.*, 2015, 112(23):7285-90.
- Deng Q, Ramskold D, Reinius B & Sandberg R. Single-cell RNA-seq reveals dynamic random monoallelic gene expression in mammalian cells. *Science*, 2014, 343:193-196.
- Ding J, Condon AE & Shah SP. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nat Commun.*, 2018, 9:2002.
- Eraslan G, Simon LM, Mircea M, Mueller NS & Theis FJ. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat Commun.*, 2019, 10(1):390.

- Fan X, Zhang X, Wu X, Guo H, Hu Y, Tang f, et al. Single-cell RNA-seq transcriptome analysis of linear and circular RNAs in mouse preimplantation embryos. *Genome Biol.*, 2015, 16:148.
- Gong W, Kwak Y, Pota P, Nakagawa NK, Garry D. DrImpute: imputing dropout events in single cell RNA sequencing data. *BMC Bioinformatics*, 2018, 19:220.
- Goolam M, Scialdone A, Graham SJ, Macaulay IC, Jedrusik A, Hupalowska A, et al. Heterogeneity in Oct4 and Sox2 targets biases cell fate in 4-cell mouse embryos. *Cell*, 2016, 165:61-74.
- Haque A, Engel J, Teichmann SA & Lönnberg T. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. *Genome Med.*, 2017, 9(1):75.
- Hashimshony T, Wagner F, Sher N & Yanai I. CEL-Seq2: sensitive highly-multiplexed single-cell RNA-Seq. *Genome Biol.*, 2016, 17:77.
- Huang M, Wang J, Torre E, Dueck H, Shaffer S, Bonasio R, et al. SAVER: gene expression recovery for single-cell RNE sequencing. *Nat Methods*, 2018, 15(7):539-542.
- Hu Q & Greene CS. Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell RNA transcriptomics. *Pac Symp Biocomput.*, 2019, 24:362-373.
- Hubert L & Arabie P. Comparing partitions. *J Classif.*, 1985, 2(1):193-218.
- Kingma DP & Ba J. Adam: A Method for Stochastic Optimization. arXiv: 1412.6980v1 [Preprint]. 2014 [cited 2014 Dec 22]. Available from: <https://arxiv.org/abs/1412.7003v1>.
- Klein AM, Mazutis L, Akartuna I, Tallapragada N, Veres A, Li V, et al. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 2015, 161(5):1187-1201.
- Kolodziejczuk AA, Kim JK, Svensson V, Marioni JC & Teichmann SA. The technology and Biology of Single-Cell RNA Sequencing. *Molecular Cell*, 2015, 58(4):610-620.
- Kolodziejczyk AA, Kim JK, Tsang JC, Ilicic T, Henriksson J, Natatajan KN, et al. Single cell RNA-sequencing of pluripotent states unlocks modular transcriptional variation. *Cell Stem Cell*, 2015, 17:471-485.
- Kvalseth TO. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 1987, 17(3):517-519.
- Li WV & Li JJ. An accurate and robust imputation method scImpute for single-cell RNA-seq data. *Nat Commun.*, 2018, 9(1):997.

- Lin P, Troup M & Ho JWK. CIDR: Ultrafast and accurate clustering through imputation for single-cell RNA-seq data. *Genome Biol.*, 2017, 18(1):59.
- Macosko EZ, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 2015, 161:1202–1214.
- Manno GL, Gyllborg D, Codeluppi S, Nishimura K, Salto C, Zeisel A, et al. Molecular Diversity of Midbrain Development in Mouse, Human, and Stem Cells. *Cell*, 2016, 167(2):566-580.
- Moon KR, Stanley JS & Burkhardt D. Manifold learning-based methods for analyzing single-cell RNA-sequencing data. *Curr Opin Syst Biol.*, 2018, 7:36-46.
- Moon KR, van Dijk D, Wang Z, Burkhardt D, Chen WS, van den Elzen A, et al. Visualizing Structure and Transitions for Biological Data Exploration. *BioRxiv* [Preprint]. 2018;bioRxiv 20378 [posted 2019 April 04]. Available from: <https://www.biorxiv.org/content/10.1101/120378v4> doi: 10.1101/120378.
- O’Leary NA, Wright MW, Brister JR, Ciufu S, Haddad D, McVeigh R, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.*, 2016, 44:733-745.
- Patro R, Duggal G, Love ML, Irizarry RA & Kingsford C. Salmon provides fast and bias-aware quantification of transcript expression. *Nat Methods*, 2017, 14:417-419.
- Pierson E & Yau C. ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biol.*, 2015, 16:241.
- Pollen AA, Nowakowski TJ, Shuga J, Wang X, Leyrat AA, Lui JH, et al., Low-coverage single-cell mRNA sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex. *Nat Biotechnol.* 2014;32:1053-1058.
- Romanov RA, Zeisel A, Bakker J, Girach F, Hellysaz A, Tomer R, et al. Molecular interrogation of hypothalamic organization reveals distinct dopamine neuronal subtypes. *Nat Neurosci.*, 2017, 20:176-188.
- Shapiro E, Biezuner & Linnarsson S. Single-cell sequencing-based technologies will revolutionize whole-organism science. *Nat Rev Genet.*, 2013, 14(9):618-30.
- Stegle O, Teichmann SA & Marioni JC. Computational and analytical challenges in single-cell transcriptomics. *Nat Rev Genet.*, 2015, 16(3):133-45.
- Talwar D, Mongia A, Sengupta D & Majumdar A. AutoImpute: Autoencoder based imputation of single-cell RNA-seq data. *Sci Rep.*, 2018, 8(1):16329.

- Usoskin D, Furlan A, Islam S, Abdo H, Lonnerberg P, Lou D, et al. Unbiased classification of sensory neuron types by large-scale single-cell RNA sequencing. *Nat Neurosci.*, 2015, 18:145-153.
- Van der Maaten L & Hinton G. Visualizing data using t-SNE. *J Mach Learn Res.*, 2008, 9:2579-2605.
- van Dijk D, Sharma R, Nainys J, Yin K, Kathail P, Carr AJ, et al. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 2018, 174(3):716-729.
- Vieth B, Ziegenhain C, Parekh S, Enard W & Hellmann I. powsimR: pow analysis for bulk and single cell RNA-seq experiment. *Bioinformatics*, 2017, 33(21):3486-3488.
- Vinh NX, Epps J & Bailey J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J Mach Learn Res.*, 2010, 11:2837-2854.
- Wagner F, Yan Y & Yanai I. K-nearest neighbor smoothing for high-throughput single-cell RNA-Seq data. *BioRxiv* [Preprint]. 2017 bioRxiv 217737 [posted 2017 Nov 21; revised 2018; cited 2018 Apr 9]. Available from: <https://www.biorxiv.org/content/10.1101/217737v3> doi: 10.1101/217737.
- Wang D & Gu J. VASC: Dimension Reduction and Visualization of Single-cell RNA-seq Data by Deep Variational Autoencoder. *Genomics Proteomics Bioinformatics*, 2018, 16(5):320-331.
- Wang J, Agarwal D, Huang M, Hu G, Zhou Z, Ye C, et al. Data denoising with transfer learning in single-cell transcriptomics. *Nat Methods*, 2019, 16,875-878.
- Wang YJ, Schug J, Won KJ, Liu C, Naji A, Avrahami D, et al. Single-Cell Transcriptome of the Human Endocrine Pancreas. *Diabetes*, 2016, 65(10):3028-3038.
- Wang Z, Gerstein M & Snyder M. RNA-Seq: revolutionary tool for transcriptomics. *Nat Rev Genet.*, 2009, 10(1):57-63.
- Way GP & Greene CS. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *Pac. Symp. Biocomput.*, 2018, 23:80–91.
- Wold S, Esbensen K & Geladi P. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*. 1987, 2:37-52
- Xin Y, Kim J, Okamoto H, Ni M, Wei Y, Adler C, et al. RNA sequencing of single human islet cells reveals type 2 diabetes genes. *Cell Metabolism*, 2016, 24:608-615.
- Yan L, Yang M, Guo H, Yang L, Wu J, Li R, et al. Single-cell RNA-Seq profiling of human preimplantation embryos and embryonic stem cells. *Nat Struct Mol Biol.*, 2013, 20:1131-1139.

- Zappia L, Phipson B & Oshlack A. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol.*, 2017, 18(1):174.
- Zeisel A, Munoz-Manchado AN, Codeluppi S, Jonnerberg P, Manno GL, Jureus A, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, 2015, 347:1138-1142.

Supporting Information

Table S1. Average NMI values for individual conditions of Splatter simulated data.

Pipeline	splat_1	splat_2	splat_3	splat_4	splat_5	splat_6	splat_7	splat_8	Mean
PCA	0.004	0.233	0.010	0.218	0.001	0.008	0.010	0.231	0.090
t-SNE	1.000	0.999	0.997	0.991	0.002	0.010	0.998	0.991	0.748
ZIFA	1.000	0.350	1.000	0.346	1.000	0.336	1.000	0.346	0.672
PHATE	1.000	0.949	1.000	0.976	0.000	0.011	1.000	0.985	0.740
CIDR	1.000	0.667	1.000	0.871	1.000	0.599	1.000	0.871	0.876
MAGIC	1.000	1.000	0.873	1.000	0.896	0.686	0.802	1.000	0.907
SAVER	1.000	1.000	1.000	1.000	1.000	0.991	1.000	1.000	0.999
SAVER-X	1.000	1.000	0.994	1.000	1.000	0.987	0.996	1.000	0.997
scImpute	0.924	0.972	1.000	1.000	0.003	0.294	1.000	1.000	0.774
DCA	0.311	0.991	0.550	0.998	0.005	0.007	0.631	0.997	0.562
autoImpute	0.000	0.011	0.175	0.040	0.004	0.015	0.341	0.053	0.080
DrImpute	1.000	0.996	1.000	0.999	0.004	0.363	1.000	1.000	0.795
LSimpute	1.000	0.999	1.000	1.000	0.006	0.342	1.000	1.000	0.793
kNN-smoothing	1.000	1.000	1.000	1.000	1.000	1.000	0.996	1.000	1.000
scRMD	1.000	0.996	1.000	1.000	0.004	0.332	1.000	1.000	0.792
DAE	0.603	0.512	1.000	0.998	0.996	0.689	1.000	0.998	0.849

Table S2. Average HOMO values for individual conditions of Splatter simulated data.

Pipeline	splat_1	splat_2	splat_3	splat_4	splat_5	splat_6	splat_7	splat_8	Mean
PCA	0.004	0.230	0.010	0.214	0.001	0.008	0.010	0.227	0.088
t-SNE	1.000	0.999	0.997	0.991	0.002	0.009	0.998	0.991	0.748
ZIFA	1.000	0.340	1.000	0.337	1.000	0.326	1.000	0.337	0.667
PHATE	1.000	0.949	1.000	0.976	0.000	0.010	1.000	0.985	0.740
CIDR	1.000	0.665	1.000	0.871	1.000	0.599	1.000	0.871	0.876
MAGIC	1.000	1.000	0.871	1.000	0.894	0.683	0.802	1.000	0.906
SAVER	1.000	1.000	1.000	1.000	1.000	0.991	1.000	1.000	0.999
SAVER-X	1.000	1.000	0.994	1.000	1.000	0.987	0.996	1.000	0.997
scImpute	0.923	0.971	1.000	1.000	0.003	0.293	1.000	1.000	0.774
DCA	0.309	0.991	0.549	0.998	0.005	0.007	0.631	0.997	0.561
autoImpute	0.000	0.009	0.174	0.040	0.004	0.015	0.341	0.053	0.080
DrImpute	1.000	0.996	1.000	0.999	0.004	0.361	1.000	1.000	0.795
LSimpute	1.000	0.999	1.000	1.000	0.006	0.340	1.000	1.000	0.793
kNN-smoothing	1.000	1.000	1.000	1.000	1.000	1.000	0.996	1.000	1.000
scRMD	1.000	0.996	1.000	1.000	0.004	0.331	1.000	1.000	0.791
DAE	0.603	0.507	1.000	0.998	0.996	0.687	1.000	0.998	0.849

Table S3. Average NMI values for individual conditions of powsimR simulated data.

Pipeline	pow_1	pow_2	pow_3	pow_4	pow_5	pow_6	pow_7	pow_8	pow_9	Mean
PCA	0.469	0.486	0.460	0.502	0.455	0.462	0.467	0.453	0.465	0.469
t-SNE	0.168	0.628	0.561	0.035	0.589	0.593	0.068	0.590	0.616	0.428
ZIFA	0.564	0.542	0.551	0.584	0.598	0.526	0.628	0.607	0.565	0.574
PHATE	0.690	0.707	0.690	0.750	0.702	0.692	0.673	0.714	0.703	0.702
CIDR	0.670	0.869	0.672	0.682	0.670	0.667	0.720	0.667	0.671	0.698
MAGIC	0.679	0.557	0.668	0.671	0.670	0.658	0.675	0.673	0.469	0.635
SAVER	0.672	0.924	0.668	0.667	0.668	0.667	0.690	0.669	0.670	0.699
SAVER-X	0.671	0.819	0.668	0.667	0.670	0.667	0.684	0.667	0.674	0.687
scImpute	0.667	0.667	0.667	0.667	0.659	0.644	0.669	0.663	0.668	0.663
DCA	0.516	0.534	0.331	0.516	0.441	0.490	0.667	0.494	0.369	0.484
autoImpute	0.609	0.590	0.517	0.597	0.636	0.426	0.564	0.653	0.678	0.586
DrImpute	0.667	0.667	0.667	0.622	0.564	0.596	0.664	0.662	0.677	0.643
LSimpute	0.667	0.669	0.667	0.622	0.669	NA	0.643	NA	NA	0.656
kNN-smoothing	0.667	0.983	0.861	0.679	0.641	0.640	0.689	0.650	0.670	0.720
scRMD	0.667	0.667	0.667	0.668	0.667	0.667	0.643	0.663	0.667	0.664
DAE	0.670	0.667	0.653	0.670	0.613	0.720	0.667	0.503	0.573	0.637

Table S4. Average HOMO values for individual conditions of powsimR simulated data.

Pipeline	pow_1	pow_2	pow_3	pow_4	pow_5	pow_6	pow_7	pow_8	pow_9	Mean
PCA	0.480	0.497	0.472	0.511	0.467	0.474	0.477	0.465	0.477	0.480
t-SNE	0.133	0.583	0.525	0.016	0.543	0.548	0.033	0.547	0.562	0.388
ZIFA	0.568	0.544	0.554	0.574	0.597	0.531	0.621	0.602	0.566	0.573
PHATE	0.677	0.667	0.667	0.667	0.667	0.667	0.667	0.667	0.660	0.667
CIDR	0.670	0.869	0.667	0.677	0.667	0.667	0.720	0.667	0.667	0.697
MAGIC	0.679	0.564	0.667	0.670	0.667	0.661	0.675	0.672	0.478	0.637
SAVER	0.667	0.924	0.667	0.667	0.667	0.667	0.667	0.667	0.666	0.695
SAVER-X	0.667	0.819	0.667	0.667	0.667	0.667	0.667	0.667	0.667	0.684
scImpute	0.667	0.667	0.667	0.667	0.658	0.645	0.669	0.662	0.667	0.663
DCA	0.525	0.540	0.337	0.525	0.451	0.496	0.667	0.502	0.375	0.491
autoImpute	0.610	0.588	0.522	0.597	0.637	0.433	0.562	0.653	0.652	0.584
DrImpute	0.667	0.667	0.667	0.619	0.562	0.599	0.663	0.662	0.667	0.641
LSimpute	0.667	0.669	0.667	0.619	0.667	NA	0.642	NA	NA	0.655
kNN-smoothing	0.667	0.983	0.864	0.667	0.641	0.640	0.682	0.649	0.667	0.718
scRMD	0.667	0.667	0.666	0.667	0.666	0.667	0.642	0.662	0.667	0.663
DAE	0.668	0.667	0.653	0.667	0.614	0.719	0.667	0.508	0.577	0.638

Table S5. Average NMI values for individual real datasets.

Pipeline	Camp	Wang	Zeisel	Darmanis	Yan	Deng	Manno	Klein
PCA	0.581	0.285	0.419	0.436	0.857	0.439	0.323	0.398
t-SNE	0.781	0.418	0.711	0.597	0.791	0.663	0.458	0.587
ZIFA	0.593	NA	0.399	0.537	0.857	0.393	0.352	0.745
PHATE	0.697	0.390	0.594	0.672	0.817	0.739	0.447	0.925
CIDR	0.384	0.373	0.431	0.461	0.714	0.764	0.286	0.698
MAGIC	0.844	0.497	0.686	0.681	0.752	0.672	0.483	0.619
SAVER	0.797	0.548	0.540	0.173	0.791	0.633	0.370	0.878
SAVER-X	0.830	0.521	0.545	0.172	0.792	0.634	0.381	0.926
scImpute	NA	NA	0.450	0.202	NA	0.513	0.339	0.649
DCA	NA	NA	0.545	0.160	NA	0.474	0.413	0.688
autoImpute	0.790	0.293	0.150	0.559	0.807	0.413	0.256	0.274
DrImpute	0.857	0.412	0.664	0.752	0.792	0.688	0.503	0.822
LSimpute	0.761	0.372	NA	0.761	0.750	0.509	NA	NA
kNN-smoothing	0.807	0.514	0.619	0.434	0.781	0.638	0.447	0.820
scRMD	0.774	0.359	0.467	0.202	0.795	0.522	0.347	0.569
DAE	0.752	0.467	0.682	0.639	0.800	0.673	0.446	0.898

Table S5. Average NMI values for individual real datasets. (Continued)

Pipeline	Romanov	Usoskin	Biase	Fan	Kołodziejczyk	Xin	Goolam	Mean
PCA	0.305	0.270	0.863	0.489	0.491	0.421	0.593	0.478
t-SNE	0.483	0.347	0.755	0.515	0.451	0.066	0.703	0.555
ZIFA	0.351	0.386	0.888	0.473	0.562	0.439	0.628	0.507
PHATE	0.443	0.579	0.742	0.445	0.784	0.511	0.740	0.635
CIDR	0.390	0.182	0.324	0.355	0.538	0.174	0.835	0.461
MAGIC	0.529	0.728	0.626	0.478	0.551	0.599	0.726	0.631
SAVER	0.430	0.370	0.840	0.457	0.385	0.571	0.459	0.549
SAVER-X	0.430	0.565	0.818	0.512	0.404	NA	0.455	0.532
scImpute	0.317	NA	NA	NA	0.214	NA	0.270	0.339
DCA	0.432	NA	NA	NA	0.262	NA	0.353	0.222
autoImpute	0.336	0.240	0.629	0.395	0.355	0.338	0.583	0.428
DrImpute	0.486	0.796	0.710	0.466	0.759	0.273	0.691	0.645
LSimpute	NA	0.372	0.754	0.525	0.226	NA	0.525	0.337
kNN-smoothing	0.441	0.676	0.831	0.501	0.300	0.626	0.552	0.599
scRMD	0.363	0.590	0.821	0.525	0.210	0.232	0.264	0.469
DAE	0.479	0.568	0.754	0.423	0.621	0.514	0.701	0.628

Table S6. Average HOMO values for individual conditions of real datasets.

Pipeline	Camp	Wang	Zeisel	Darmanis	Yan	Deng	Manno	Klein
PCA	0.587	0.316	0.456	0.456	0.831	0.450	0.331	0.396
t-SNE	0.783	0.467	0.778	0.634	0.805	0.722	0.454	0.551
ZIFA	0.594	NA	0.440	0.594	0.831	0.407	0.361	0.753
PHATE	0.697	0.416	0.645	0.697	0.805	0.765	0.455	0.921
CIDR	0.340	0.380	0.450	0.487	0.608	0.798	0.290	0.694
MAGIC	0.839	0.556	0.753	0.722	0.753	0.744	0.502	0.633
SAVER	0.798	0.597	0.596	0.184	0.771	0.702	0.385	0.882
SAVER-X	0.831	0.579	0.603	0.182	0.791	0.703	0.397	0.927
scImpute	NA	NA	0.496	0.213	NA	0.567	0.353	0.657
DCA	NA	NA	0.602	0.170	NA	0.526	0.431	0.694
autoImpute	0.767	0.322	0.142	0.583	0.792	0.457	0.243	0.276
DrImpute	0.843	0.460	0.712	0.773	0.805	0.744	0.492	0.778
LSimpute	0.762	0.453	NA	0.213	0.758	0.560	NA	NA
kNN-smoothing	0.807	0.574	0.681	0.457	0.805	0.692	0.464	0.824
scRMD	0.777	0.401	0.516	0.214	0.805	0.573	0.362	0.582
DAE	0.755	0.522	0.753	0.676	0.802	0.746	0.466	0.898

Table S7. Average HOMO values for individual real datasets (Continued)

Pipeline	Romanov	Usoskin	Biase	Fan	Kołodziejczyk	Xin	Goolam	Mean
PCA	0.330	0.273	0.842	0.492	0.491	0.537	0.658	0.497
t-SNE	0.518	0.318	0.739	0.528	0.456	0.071	0.790	0.574
ZIFA	0.379	0.392	0.850	0.475	0.569	0.561	0.637	0.521
PHATE	0.481	0.591	0.746	0.441	0.793	0.643	0.757	0.645
CIDR	0.423	0.175	0.248	0.343	0.528	0.202	0.922	0.459
MAGIC	0.581	0.731	0.648	0.487	0.553	0.771	0.812	0.672
SAVER	0.476	0.280	0.851	0.453	0.384	0.600	0.513	0.565
SAVER-X	0.475	0.480	0.834	0.523	0.403	NA	0.508	0.549
scImpute	0.349	NA	NA	NA	0.211	NA	0.300	0.210
DCA	0.477	NA	NA	NA	0.260	NA	0.398	0.237
autoImpute	0.344	0.244	0.624	0.402	0.357	0.434	0.653	0.443
DrImpute	0.514	0.799	0.695	0.455	0.769	0.227	0.775	0.656
LSimpute	NA	0.342	0.763	0.532	0.223	NA	0.600	0.347
kNN-smoothing	0.488	0.689	0.842	0.512	0.299	0.805	0.600	0.636
scRMD	0.401	0.604	0.836	0.529	0.207	0.269	0.294	0.491
DAE	0.527	0.578	0.774	0.432	0.611	0.670	0.784	0.667

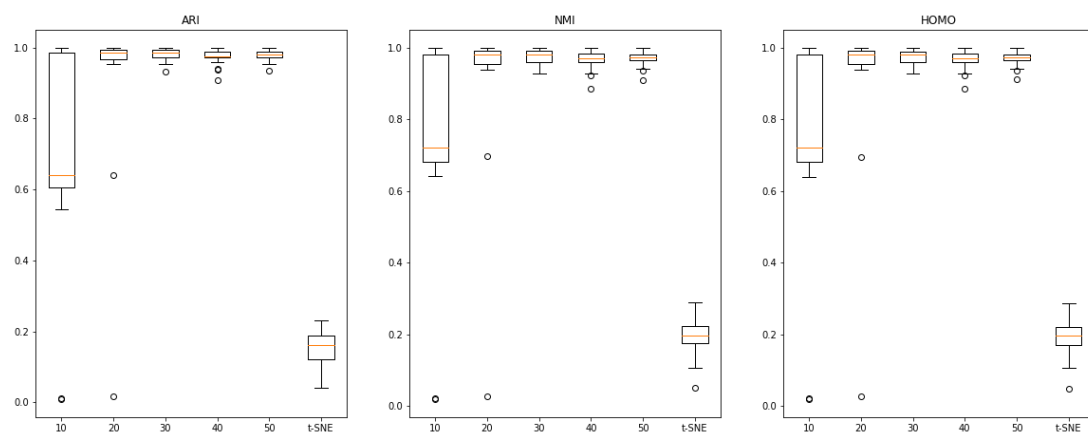


Figure S1. Best latent dimension of DAE model.

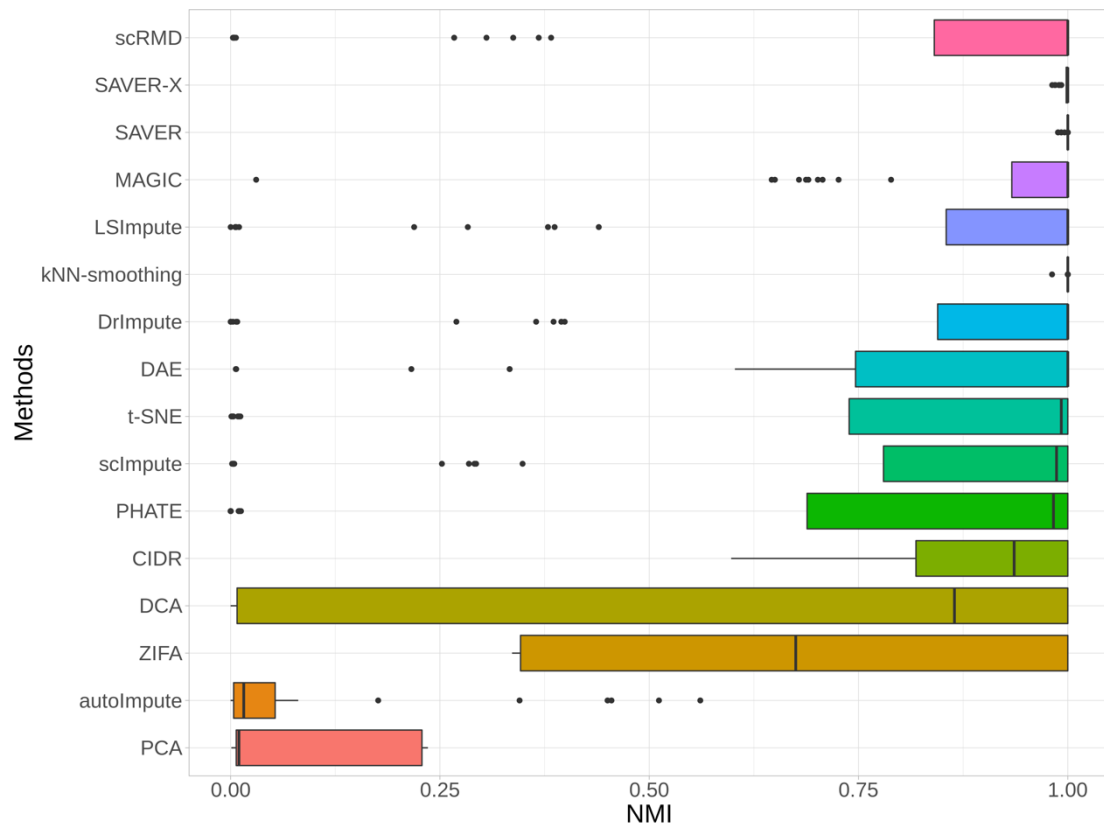


Figure S2. NMI boxplot of Splatter simulated data

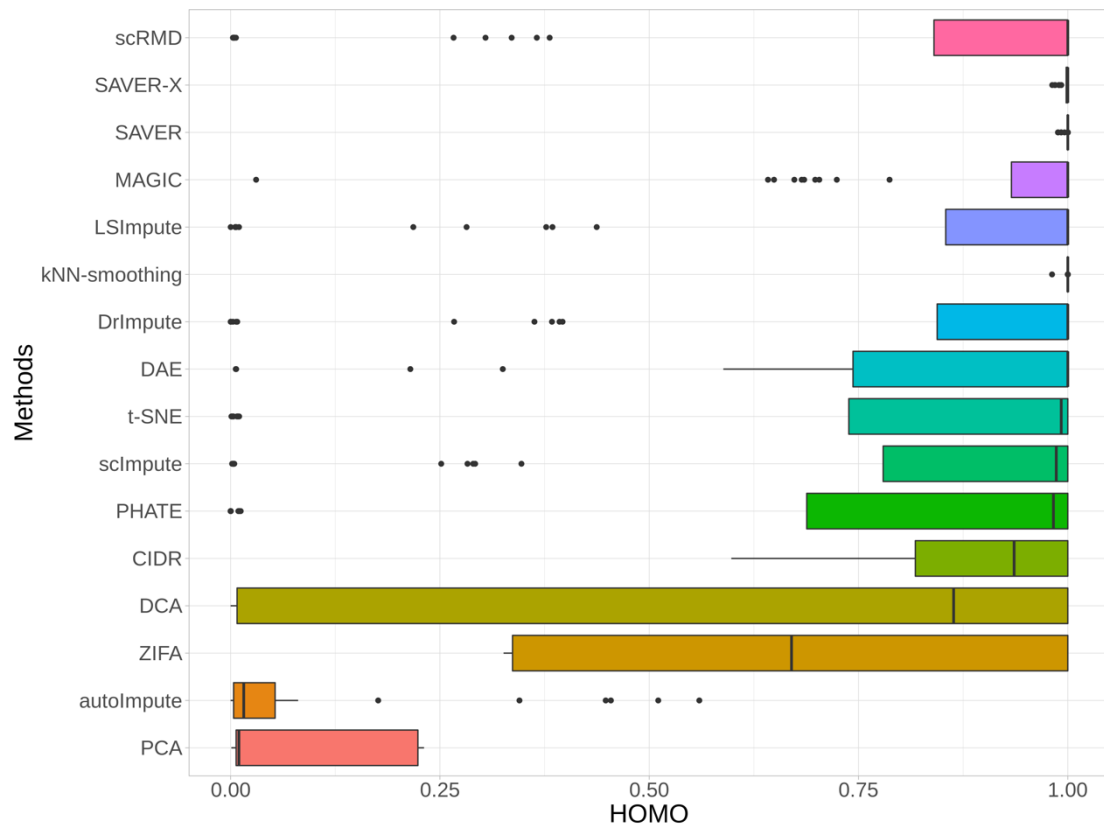


Figure S3. HOMO boxplot of Splatter simulated data

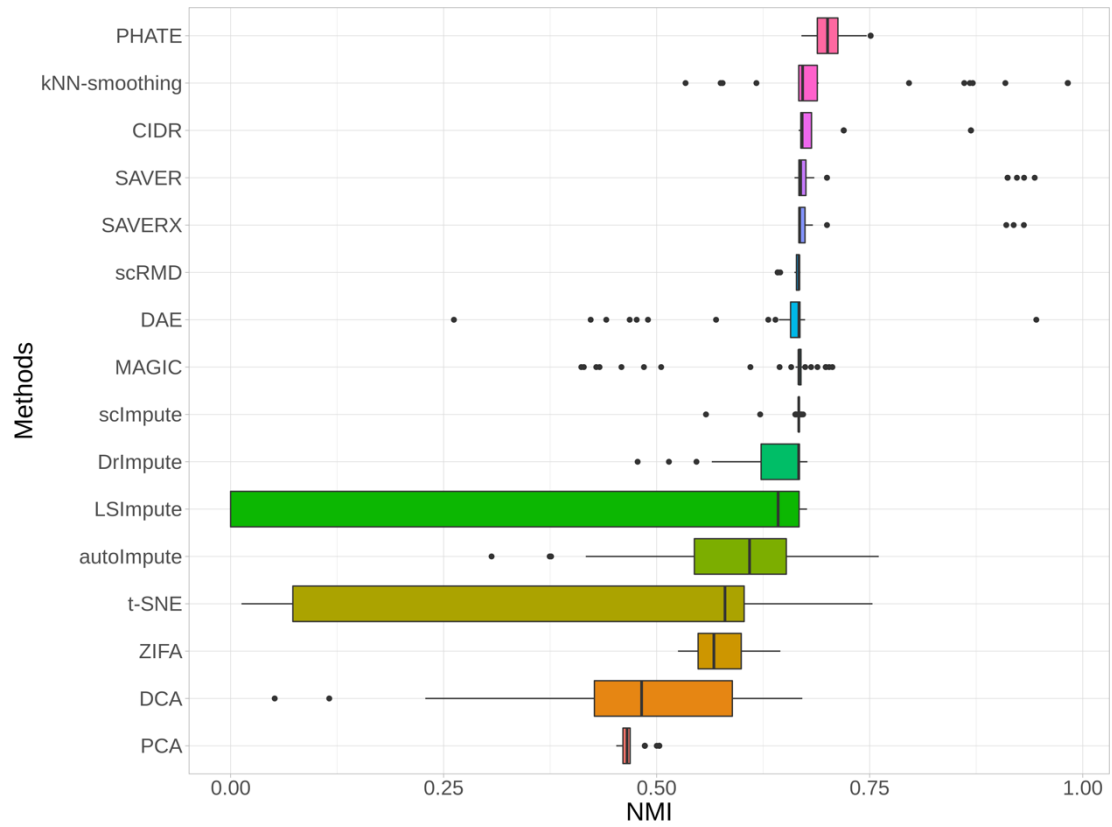


Figure S4. NMI boxplot of powsimR simulated data

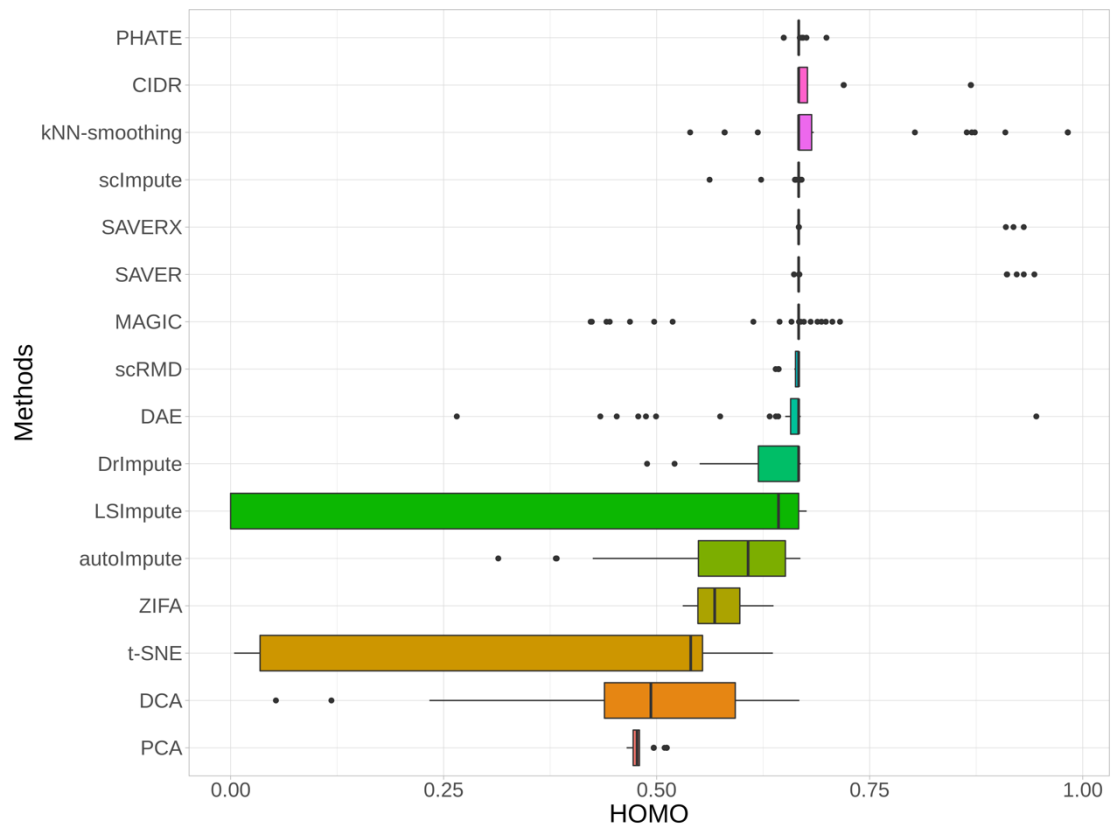


Figure S5. HOMO boxplot of powsimR simulated data

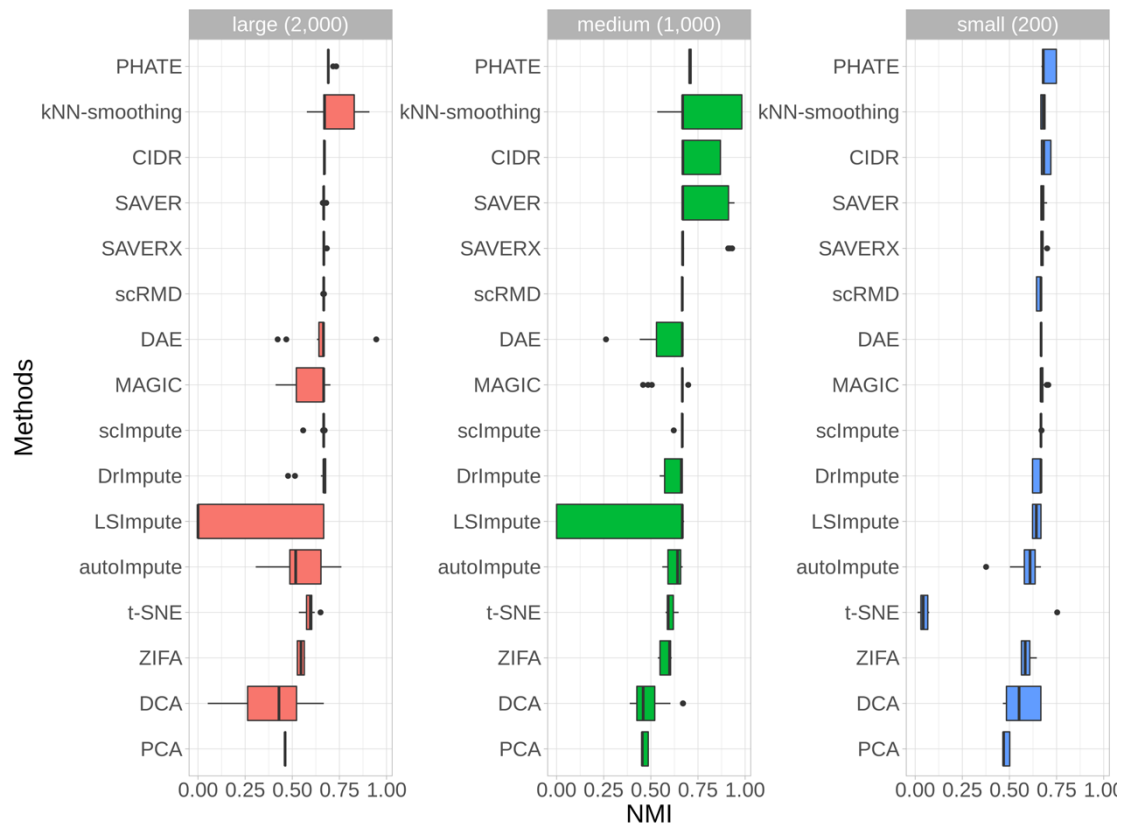


Figure S6. NMI boxplot of powsimR simulated data (different sample sizes)

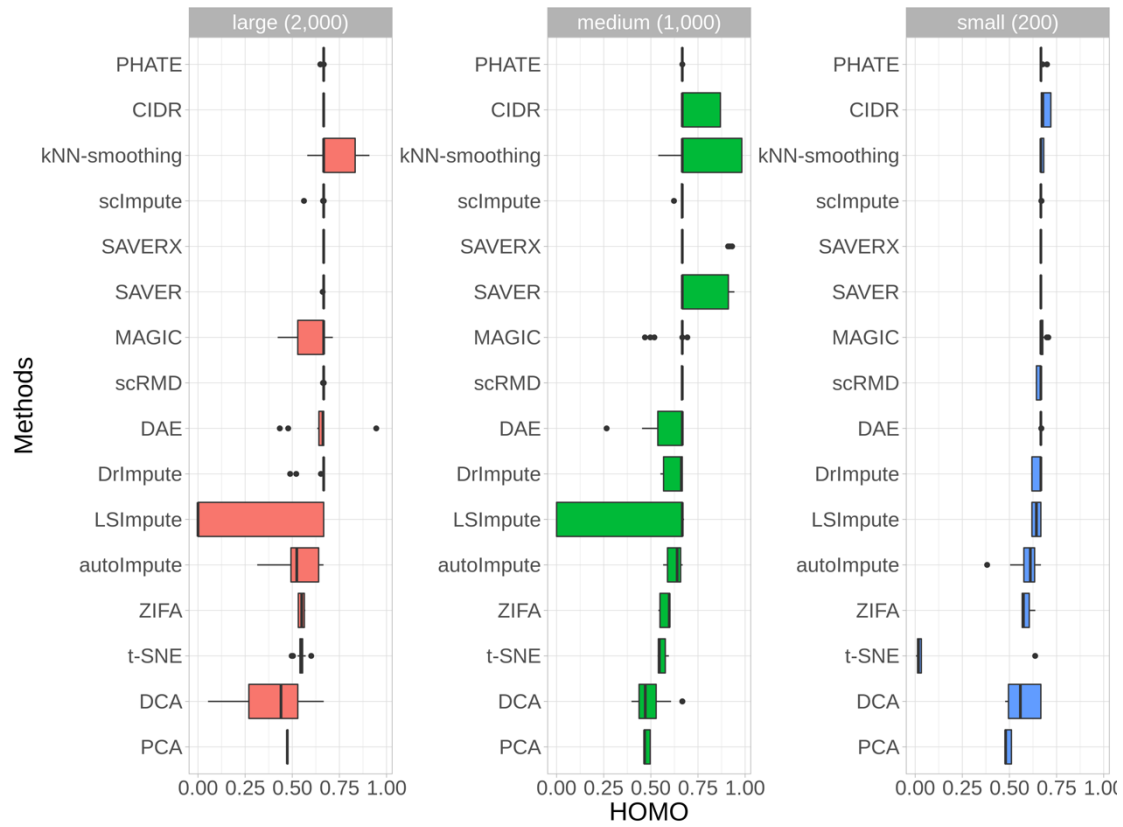


Figure S7. HOMO boxplot of powsimR simulated data (different sample sizes)

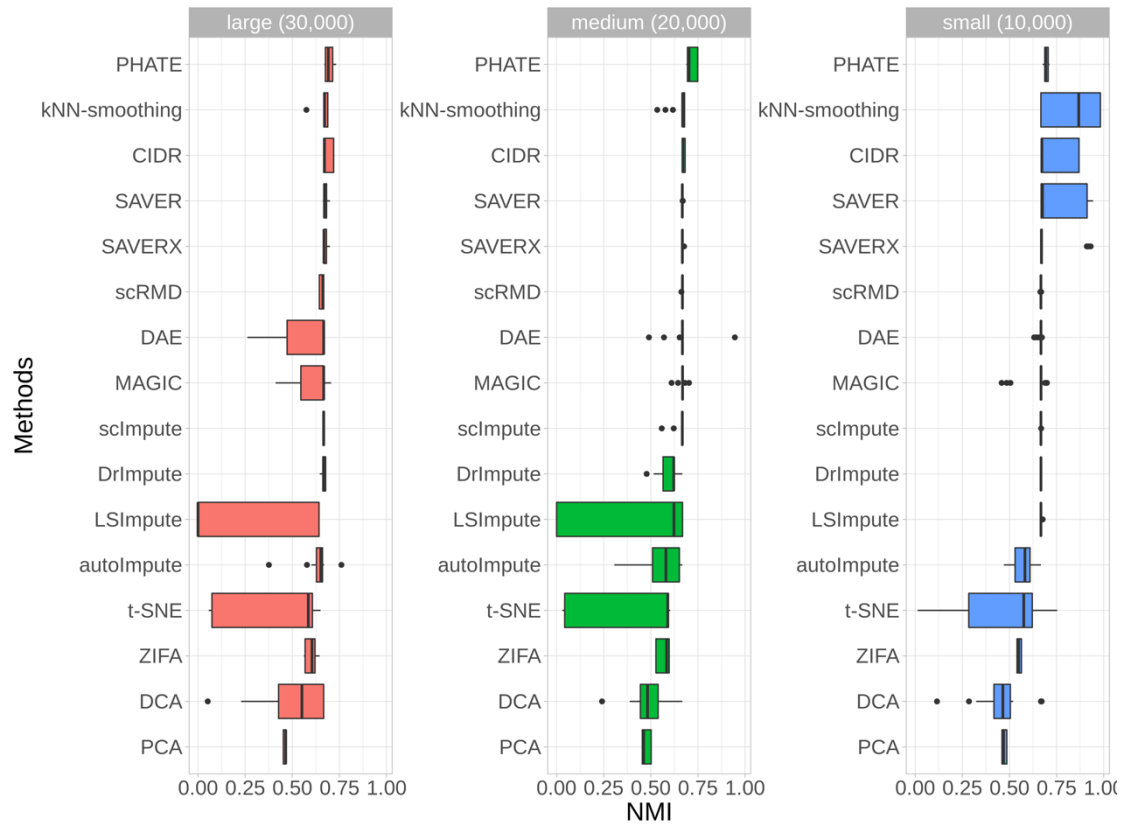


Figure S8. NMI boxplot of powsimR simulated data (different gens numbers)

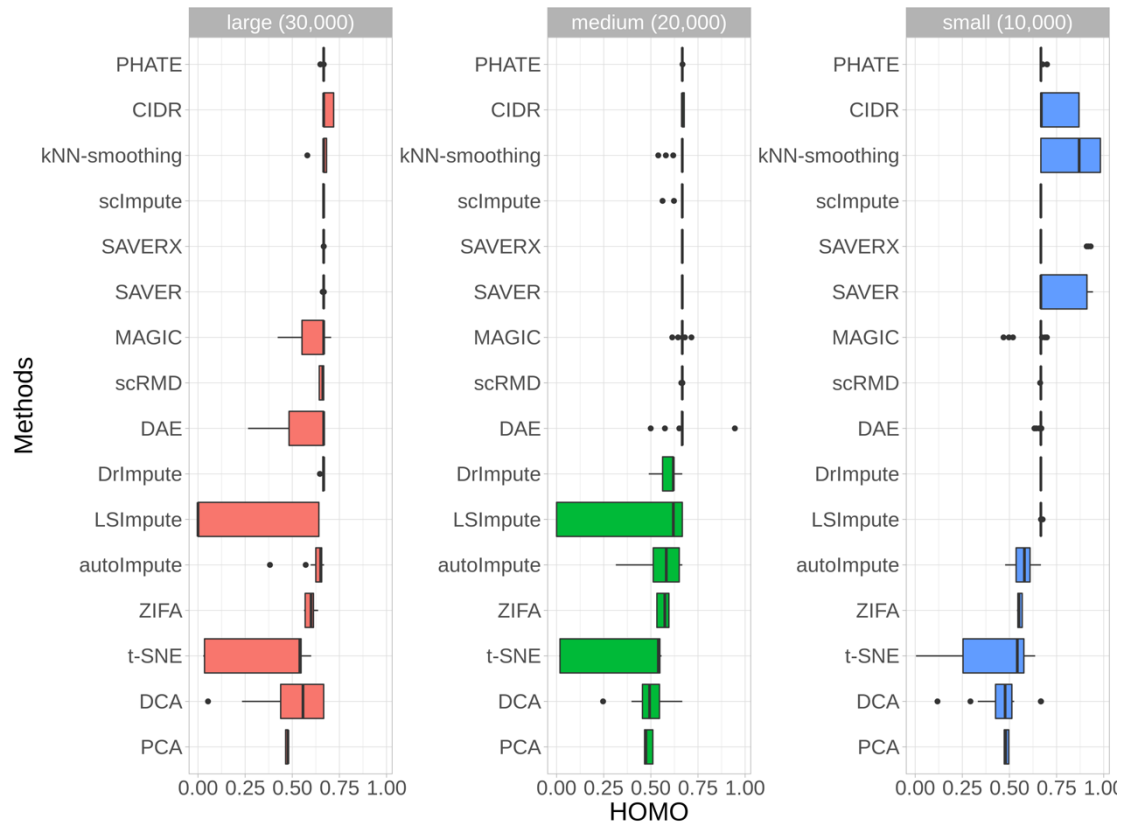


Figure S9. HOMO boxplot of powsimR simulated data (different gens numbers)

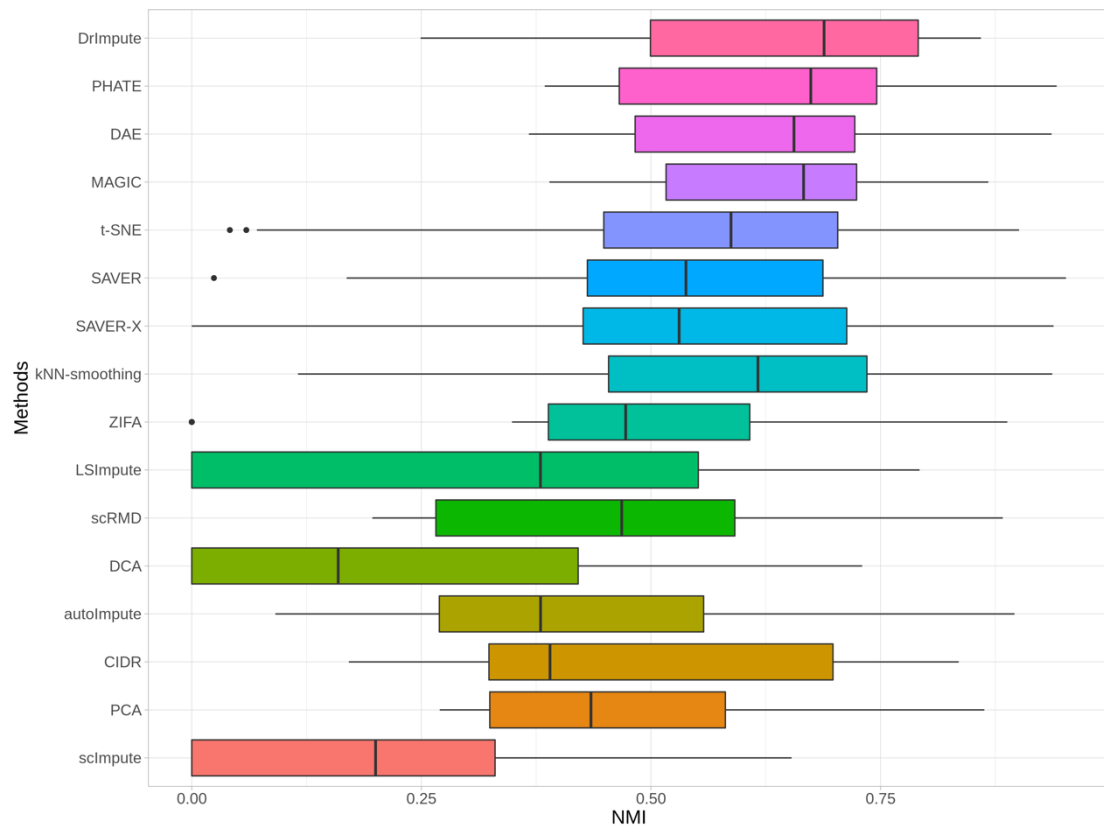


Figure S10. NMI boxplot of real data

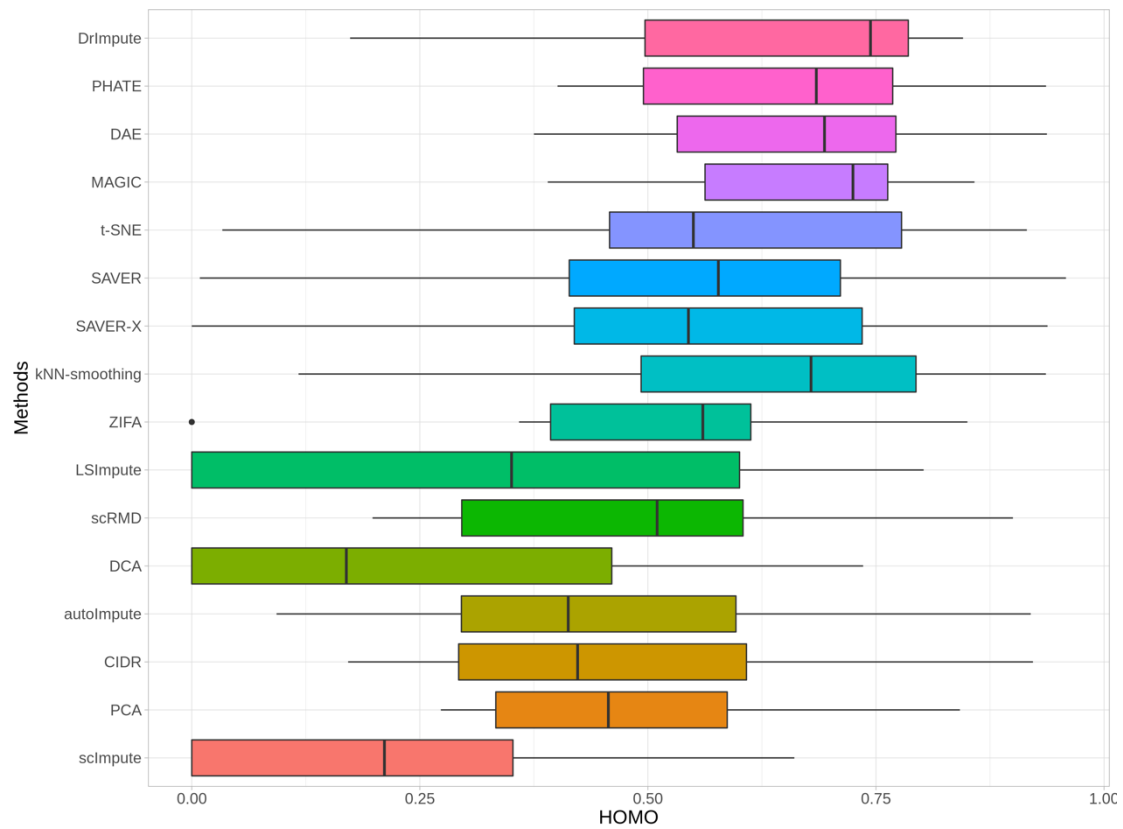


Figure S11. HOMO boxplot of real data

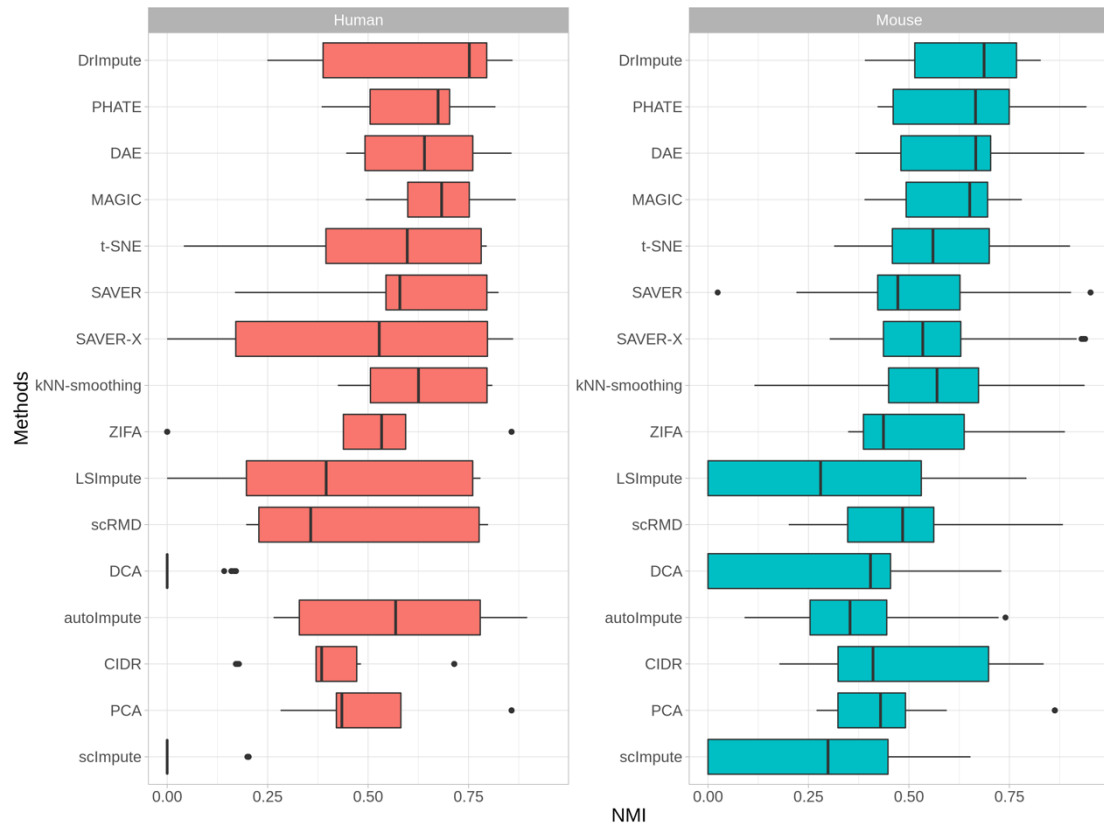


Figure S12. NMI boxplot of real data (different species)

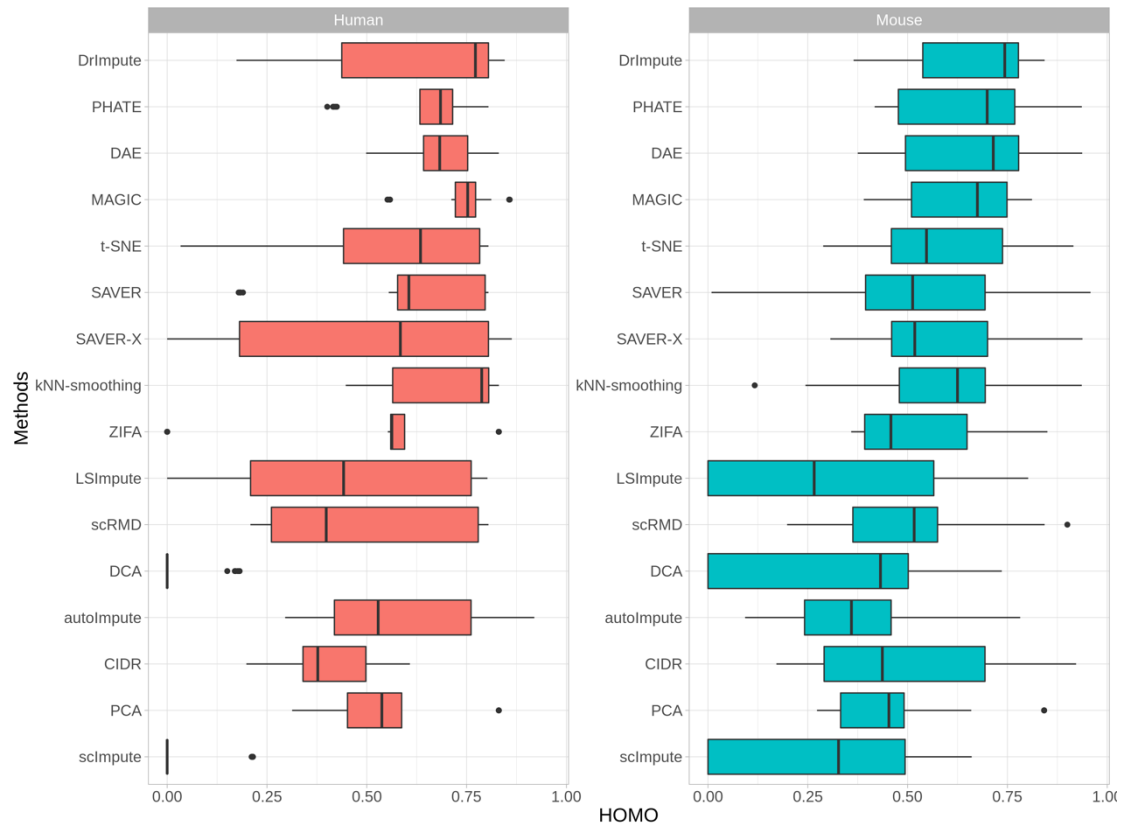


Figure S13. HOMO boxplot of real data (different species)

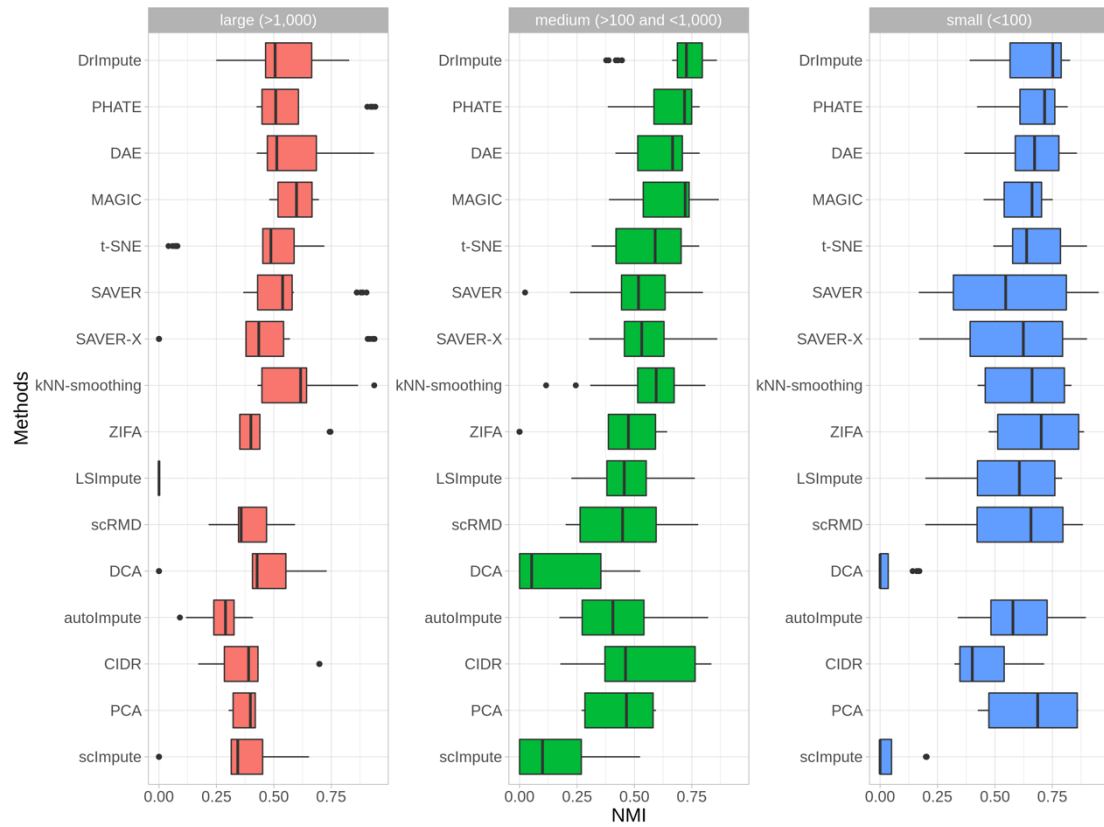


Figure S14. NMI boxplot of real data (different sample sizes)

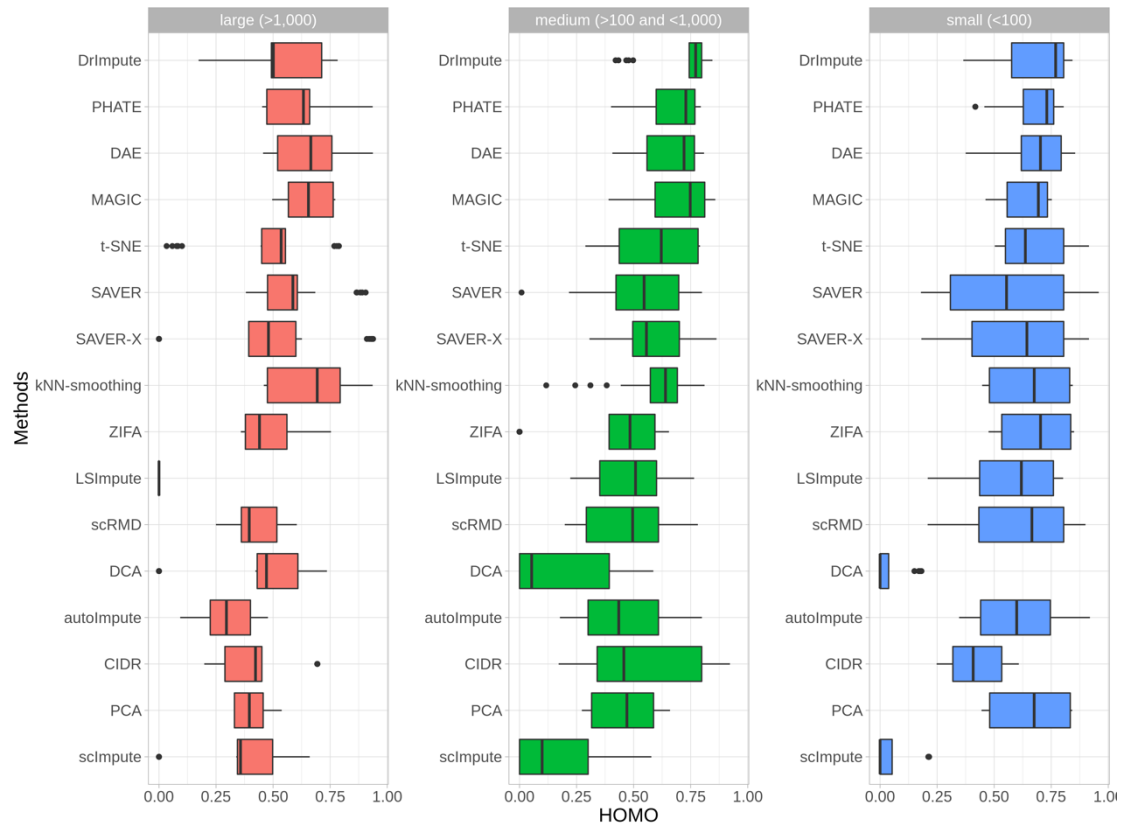


Figure S15. HOMO boxplot of real data (different sample sizes)

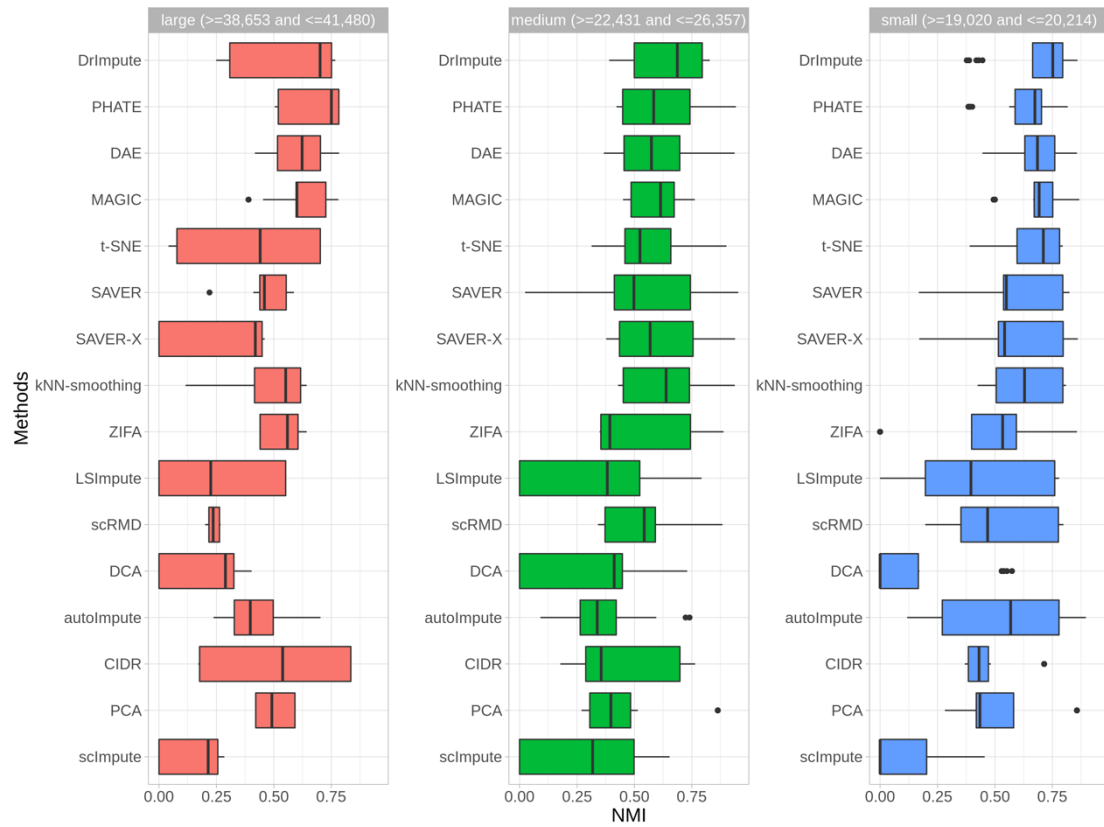


Figure S16. NMI boxplot of real data (different gens numbers)

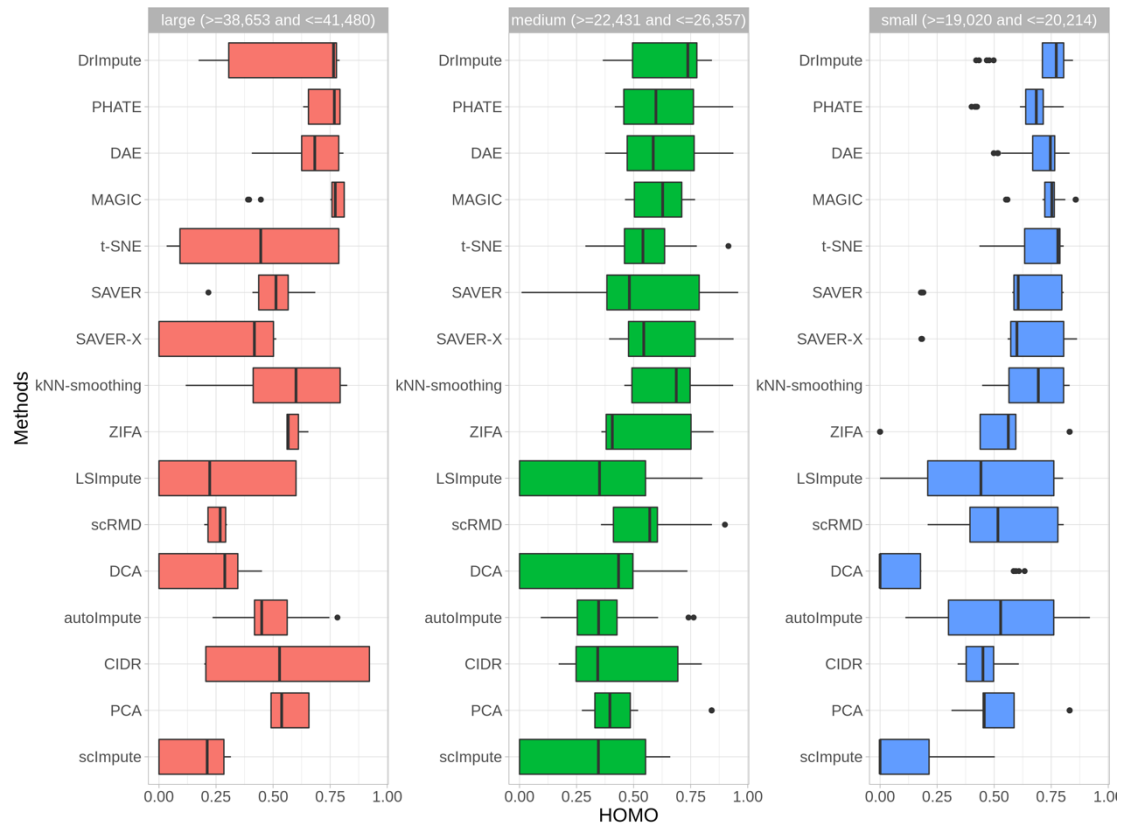


Figure S17. HOMO boxplot of real data (different gens numbers)

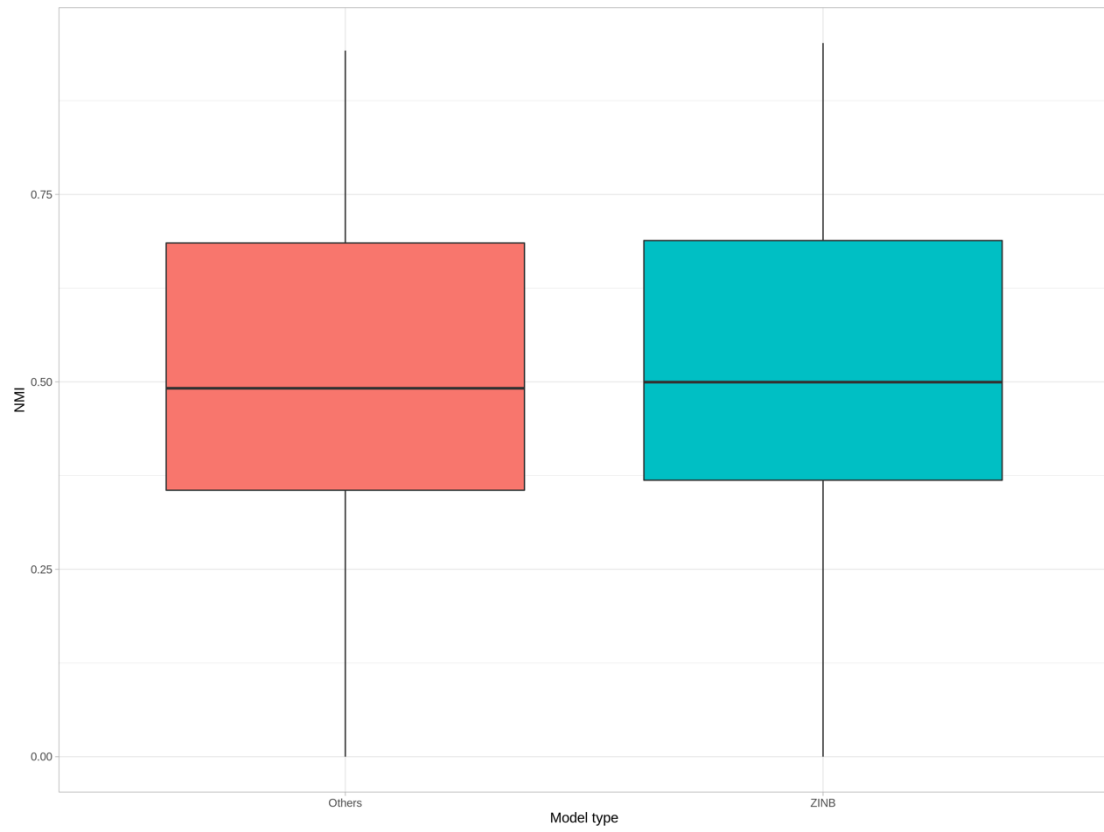


Figure S18. NMI boxplot of real data (different models)

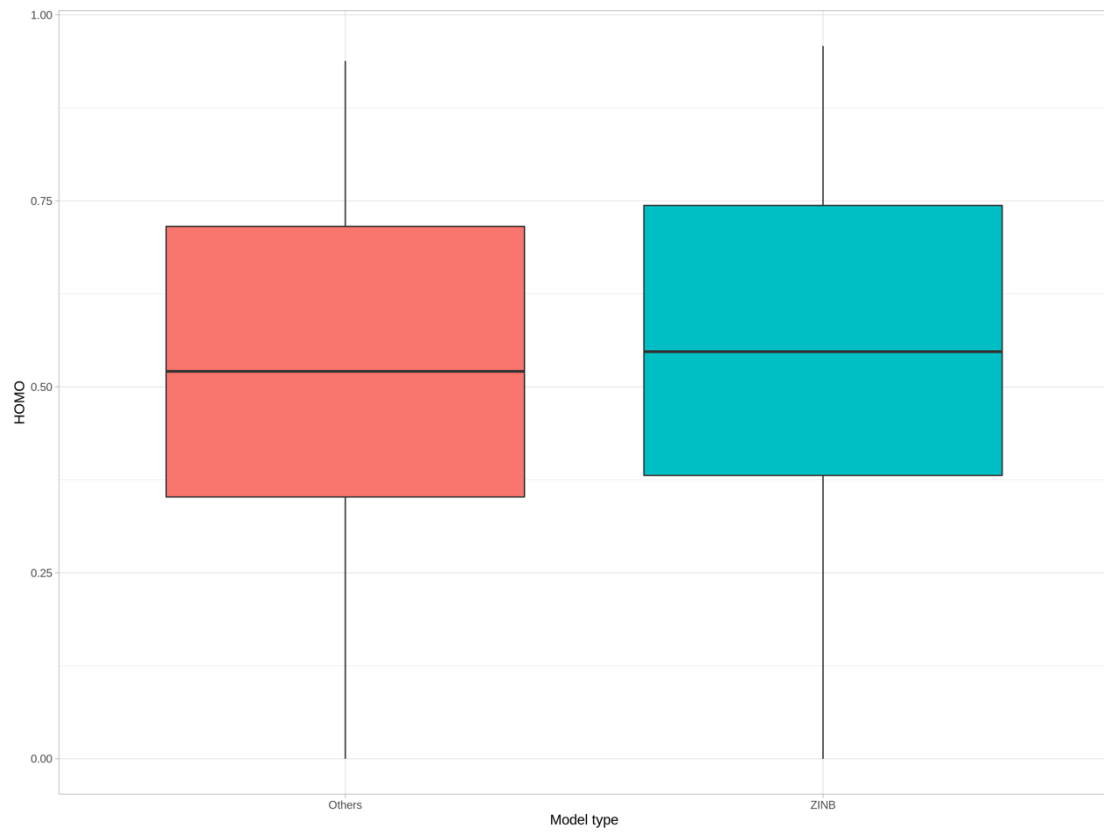


Figure S19. HOMO boxplot of real data (different models)