HiCImpute: An R Package Implementing "Bayesian Hierarchical Model for Identifying Structural Zeros and Enhancing Single-cell Hi-C Data"

Qing Xie, Shili Lin

8/16/2021

1.Introduction

Single-cell Hi-C (scHi-C) techniques enable one to study between-cell variability using long-range interaction data. However, single-cell Hi-C data typically suffer from sparsity, that is, the existence of excess zeros due to insufficient sequencing depth. Differentiating between structural zeros and sampling zeros, and accurately imputing the dropout values are important, since correct inference would improve downstream analyses such as clustering and discovery of subtypes.

HiCImpute implements a Bayesian hierarchical model that goes beyond data quality improvement by also identifying observed zeros that are in fact structural zeros (xie et al. 2021). HiCImpute takes spatial dependencies of scHi-C 2D data structure into account while also borrowing information from similar single-cells and bulk data, when such are available.

In this package, we provide the following main functions:

- **MCMCImpute**: This is the flagship function of HiCImpute. It identifies structural zeros and imputes sampling zeros under a Bayesian framework. The output can be used to facilitate downstream analysis such as clustering, subtype discovery, or 3D structure construction.
- **scHiC_assess**: This function provides imputed data quality assessment based on several criteria, depending on whether the ground truth is known (for simulated data) or not (for real data).
- **scHiC_simulate**: This function simulates single-cell Hi-C data based on a given chromatin structure represented by 3D coordinates.

We will illustrate the usage of these functions in the following sections.

2. MCMCImpute

```
MCMCImpute(scHiC, bulk, startval, n, epsilon1, epsilon2, mc.cores, cutoff, niter, burnin)
```

2.1 Input data and format

The main input data are in **scHiC**.

scHiC can take three types of formats. The preferred format is a single-cell matrix with each column being a vector of the upper triangular matrix without including the diagonal entries of the 2D matrix of a single-cell. Another types of formats are a list with each element being a 2D single-cell contact matrix, or a 3D $(n \times n \times k)$ array that has k matrices of dimension $n \times n$. HiCImpute automatically transforms these two types of input into a matrix with each column being the vector of upper triangular matrix of a single-cell. For a single-cell matrix of size $n \times n$, the length of the vector should be $n \times (n-1)/2$. We only need the upper triangular matrix because the Hi-C matrix are symmetrical.

Here is an example for the single-cell matrix in preferred format. In this example, there are 100 single-cells, and each column is a vector of the upper triangular entries of each single-cell. Since this K562_T1_7k is of dimension 61×61, so each single-cell has a vector of length $61 \times 60/2 = 1830$.

```
options(digits = 2)
library(HiCImpute)
data("K562_T1_4k")
K562_T1_4k[1:10,1:10]
#>       c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_10
#>  [1,]   2   3   3   5   4   4   4   4   4    4
#>  [2,]   2   2   2   2   2   2   2   2   2    1
#>  [3,]   3   3   2   3   1   1   2   3   2    3
#>  [4,]   1   0   1   0   1   1   1   1   0    0
#>  [5,]   2   1   2   1   2   2   2   2   2    2
#>  [6,]   6   4   6   5   4   4   6   4   4    4
#>  [7,]   1   1   0   1   1   1   0   1   1    1
#>  [8,]   2   4   3   3   3   2   3   4   2    1
#>  [9,]   4   6   8   8   6   8   6   7   6    7
#> [10,]   7   8  15  13  12  12  15  13   9   15
```

The following are examples of another two types of input format. The first one is a list with each element being a 2D single-cell contact matrix. In this example, the list has length of 100 and each list element is a 61×61 single-cell matrix. Another format is a 3D ($61 \times 61 \times 100$) array that has 100 matrices of dimension 61×61.

```
options(digits = 2)
data("K562_T1_4k_list")
K562_T1_4k_list[[1]][1:10,1:10]
#>          loci_ 1 loci_ 2 loci_ 3 loci_ 4 loci_ 5 loci_ 6 loci_ 7 loci_ 8
#> loci_ 1        0       2       2       1       1       1       2       2
#> loci_ 2        2       0       3       2       2       2       2       2
#> loci_ 3        2       3       0       6       4       2       3       0
#> loci_ 4        1       2       6       0       7       1       1       1
#> loci_ 5        1       2       4       7       0       3       2       1
#> loci_ 6        1       2       2       1       3       0       5       3
#> loci_ 7        2       2       3       1       2       5       0       5
#> loci_ 8        2       2       0       1       1       3       5       0
#> loci_ 9        4       4       3       2       3       4       6       7
#> loci_ 10       3       4       1       1       2       2       4       7
#>          loci_ 9 loci_ 10
#> loci_ 1        4        3
#> loci_ 2        4        4
#> loci_ 3        3        1
#> loci_ 4        2        1
#> loci_ 5        3        2
#> loci_ 6        4        2
#> loci_ 7        6        4
#> loci_ 8        7        7
#> loci_ 9        0        7
#> loci_ 10       7        0
```

```
data("K562_T1_4k_3D_array")
K562_T1_4k_3D_array[1:10,1:10,1]
```

```
#>        [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#>   [1,]    0    2    2    1    1    1    2    2    4     3
#>   [2,]    2    0    3    2    2    2    2    2    4     4
#>   [3,]    2    3    0    6    4    2    3    0    3     1
#>   [4,]    1    2    6    0    7    1    1    1    2     1
#>   [5,]    1    2    4    7    0    3    2    1    3     2
#>   [6,]    1    2    2    1    3    0    5    3    4     2
#>   [7,]    2    2    3    1    2    5    0    5    6     4
#>   [8,]    2    2    0    1    1    3    5    0    7     7
#>   [9,]    4    4    3    2    3    4    6    7    0     7
#>  [10,]    3    4    1    1    2    2    4    7    7     0
```

bulk can take two types of formats. A 2D bulk matrix of dimension $n \times n$ or a vector of the upper triangular entries of 2D bulk matrix. It can provide information for priors settings. If bulk data is not available, simply set it to be NULL, and MCMCImpute will sum up the single-cells to construct a bulk data.

startval is the starting value for the vector of parameters $\Theta = (\alpha, \mu^{\gamma}, \beta, \mu, a, \delta, b, \pi, s, \mu_1, \cdots, \mu_K)$. See Xie et al. for the details of these parameters. The default value is as set in the function.

n is the dimension of the 2D matrix.

epsilon1 is the range size of δ that is used to monitor the prior mean of π_{ij}, the probability that the pair (i, j) do not interact. The default value of ϵ_1 is 0.5.

epsilon2 is the range size of B that is used to monitor the prior mean of μ_{ij}, the intensity of interaction between pair (i, j). The default value of ϵ_2 is 5.

mc.cores is the number of cores to be used in mclapply function that can parallelly impute the matrix. The default value is 1 (no parallelization), but the users is advised to a higher number to increase computational speed if their computer has parallel computing capability.

cutoff is the threshold of π_{ij} that is used to define structural zeros. The default value is 0.5. That is, if the probability of being a SZ is greater than 0.5, then the pair (i, j) are labelled as not interacting due to underlying biological mechanism.

niter is the number of iterations for the MCMC run. Default is 30000.

burnin is the number of burn-in iteration. Default is 5000.

2.2 Use of MCMCImpute function

Here is an example for the use of MCMCImpute on the simulated data:

```
# data("K562_T1_7k")
# data("K562_bulk")
# scHiC=K562_T1_7k
# set.seed(1234)
# T1_7k_result=MCMCImpute(scHiC=K562_T1_7k,bulk=K562_bulk,
# startval=c(100,100,10,8,10,0.1,900,0.2,0,replicate(dim(scHiC)[2],8)),n=61,
# mc.cores = 1,cutoff=0.5, niter=2000,burnin=500)
```

For real data analysis, we use the first scHi-C dataset in Xie et al., 30 loci on chromosome1 of 32 cells (14 GM cells and 18 PBMC cells) from GSE117874, as an example. Since we know the type of each cell, we use the following functions to imputed two types of cells separately and then combine the imputed values.

```
#data("GSE117874_chr1_wo_diag")

# single=GSE117874_chr1_wo_diag[,1:14]
# set.seed(1234)
# GSE117874_GM=MCMCImpute(niter=100000,burnin=5000,single=GSE117874_chr1_wo_diag[,1:14],
#bulk=apply(single,1,sum),startval=c(100,100,10,8,10,0.1,900,0.2,0,replicate(ncol(single),8)),
#n=30,mc.cores = 1,cutoff=0.5)
#
# single=GSE117874_chr1_wo_diag[,15:32]
# set.seed(1234)
# GSE117874_PBMC=MCMCImpute(niter=100000,burnin=5000,single=GSE117874_chr1_wo_diag[,15:32],
#bulk=apply(single,1,sum),startval=c(100,100,10,8,10,0.1,900,0.2,0,replicate(ncol(single),8)),
#n=30,mc.cores = 1,cutoff=0.5)

#GSE117874_imp=cbind(GSE117874_GM$IMP1, GSE117874_PBMC$IMP1)
```

The output of MCMCImpute is a list of posterior means of the SZ probabilities, the $(\hat{\pi}_{ij})_{n\times n}$ matrix the imputed data before zeroing out the SZs (Impute_All), and imputed data zeroing out the SZ entries (Impute_SZ).

```
data("T1_4k_result")
T1_4k_imp=T1_4k_result$Impute_SZ
T1_4k_result$SZ[1:10,1:10]
#>       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#>  [1,]    0 0.25 0.27 0.37 0.35 0.35 0.28 0.26 0.26  0.25
#>  [2,]    0 0.00 0.27 0.26 0.25 0.27 0.26 0.27 0.25  0.25
#>  [3,]    0 0.00 0.00 0.25 0.25 0.26 0.25 0.28 0.25  0.27
#>  [4,]    0 0.00 0.00 0.00 0.25 0.29 0.35 0.32 0.26  0.27
#>  [5,]    0 0.00 0.00 0.00 0.00 0.26 0.26 0.33 0.25  0.28
#>  [6,]    0 0.00 0.00 0.00 0.00 0.00 0.25 0.25 0.25  0.25
#>  [7,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.25  0.25
#>  [8,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25  0.25
#>  [9,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.25
#> [10,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00
```

```
dim(T1_4k_result$Impute_All)
#> [1] 1830  100
```

```
dim(T1_4k_result$Impute_SZ)
#> [1] 1830  100
```

2.3 Assessing HiCImpute imputation results

`scHiC_assess(scHiC, expected = NULL, result = NULL, imputed = NULL, n, cell_type, dims = 2, perplexity = 10, seed = 1000, kmeans = TRUE, ncenters = 2)`

scHiC is the observed data. It can take three types of formats. The preferred format is a single-cell matrix with each column being a vector of the upper triangular matrix without including the diagonal entries of the 2D matrix of a single-cell. Another types of formats are a list with each element being a 2D single-cell contact matrix, or a 3D $(n \times n \times k)$ array that has k matrices of dimension $n \times n$. HiCImpute automatically transforms these two types of input into a matrix with each column being the vector of upper triangular matrix of a single-cell.

expected is the underline true counts of the simulated data. For real data analysis, just set it as NULL.

result is the output of **MCMCImpute**. This is only for simulated data. For real data, set it as NULL.

imputed The imputed data that has the same dimension as the observed data. This is needed for real data analysis. For simulated data, set it as NULL.

cell_index indicates which single-cell is used to generate heatmaps and scatterplot. The default is 1.

n is the dimension of 2D matrix.

cell_type is a vector of underlying cell type. This is only for real data analysis.

dims is the dimension of t-SNE visualization. The default is 2.

perplexity = 10 is the perplexity parameter of t-SNE. Should satisfy $3 \times perplexity < nrow(X) - 1$.

seed is the random seed for generating t-SNE data.

kmeans is a logical parameter. Default is TRUE. If TRUE, apply K-means clustering on the t-SNE data.

ncenters is the number of centers in K-means clustering analysis. This is only needed if **K-means** is TRUE.

scHiC_assess analyzes both simulated and real datasets, depending on the inputs (if expected = NULL) of the functions. We illustrates the usage of these two cases separately.

2.3.1 Simulation study

If the data are simulated and the underlying expected values are also provided, then *scHiC_assess* firstly provides the following plots to visualize the imputation results.

- *scHiC_hm* visualizes the observed, expected, and imputed single-cell data as 2D heatmaps for the first single cell. But users can choose to visualize other cells as well.
- *scHiC_ROC* draws an ROC (Receiver operating characteristic) curve to study the interplay between PTSZ and PTDO when setting different thresholds for calling an observed zero to be an SZ.
- *SEVI* draws a scatterplot of the expected versus imputed values of the first single-cell. This serves as a visualization tool to directly assess whether dropouts are correctly recovered and accurately imputed for one single cell.

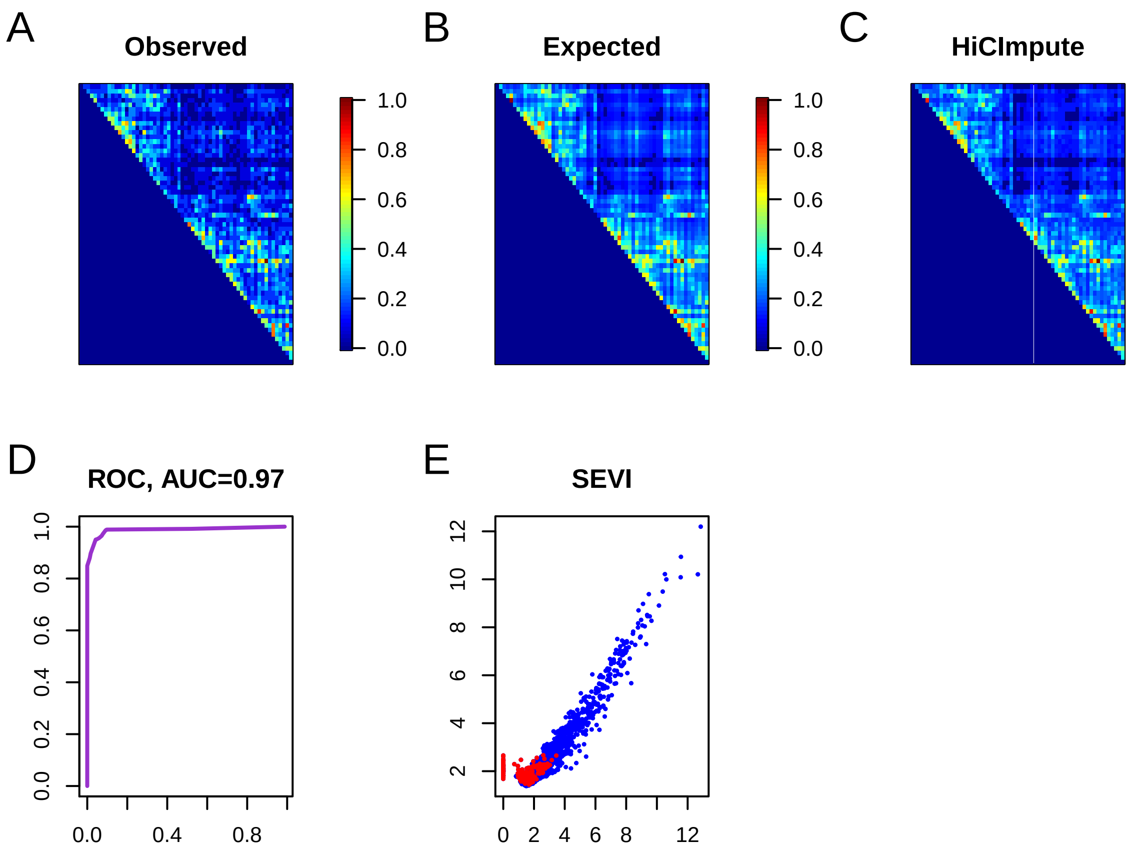
Except for these plots, scHiC_assess also summarizes the imputation accuracy in terms of the following measurements used in the paper (Xie et al. 2021).

- *PTSZ*: Proportion of true structural zeros. This is defined as the proportion of underlying structural zeros that are correctly identified as such by a method.
- *PTDO*: Proportion of true dropouts. This is defined as the proportion of underlying sam- pling zeros (due to insufficient sequencing depths) that are correctly identified as such by a method.
- *CIEZ*: Mean correlation between the imputed and expected counts for only the observed zeros.
- *CIEA*: Mean correlation between the imputed and expected counts for all observed values.
- *AEOZ*: Absolute errors for observed zeros. Unlike AEOA that considers all observed, this measure only considers observed zeros. This measure provides a more focused evaluation on correct identification oof structural zeros and accuracy of imputing dropouts.
- *AEOA*: Absolute errors for all observed data. This is defined as the imputed value minus the expected value for all observed data. This measure is to gage how well the imputed values can approximate its average underlying true values.

To gauranttee that the PTSZ is at a desired level, scHiC_assess calculates PTDO when PTSZ is fixed to be 0.95. The following is an example output of *scHiC_assess*.

The numerical summary includes the mean and standard deviation of the 6 measurements that had been used in the paper, the PTDO when PTSZ is fixed to be 0.95, and the following plots. **A-C** are the heatmaps of observed, expected, and imputed single-cell of the first cell of K562_T1_4k, where we can see that HiCImpute is able to denoise and recover the underlying structure very well. **D** is the ROC curve of the K562_T1_4k simulated data, where we can see the ROC curve goes up to 1 pretty fast. **E** is the scatterplot of the imputed versus expected values for the first K562_T1_4k single-cell, with the red points being the observed zeros. It illustrates a high correlation of the imputed and expected values.

```
data("K562_T1_4k_true")
options(digits = 2)
scHiC_assess(scHiC = K562_T1_4k, expected = K562_T1_4k_true,
                            result = T1_4k_result, n=61)
#> $summary_mean
#>   PTSZ PTDO AEOZ AEOA CIEZ CIEA
#> 1    1 0.71 0.43 0.51 0.84 0.96
#>
#> $summary_se
#>   PTSZ  PTDO AEOZ AEOA CIEZ   CIEA
#> 1    0 0.023 0.44 0.43 0.01 0.0021
#>
#> $PTSZ_95
#>   PTDO    SD2 thresh
#> 1 0.95 0.0086   0.66
```



2.3.2 Real data analysis

For a real dataset, the underlying true values are not provided; therefore, **scHiC_assess** provides the following visualization tools and Kmeans clustering analysis.

- *Boxplot* illustrates boxplot of correlations between imputed and observed values on nonzero observations.
- *SOVI* draws scatterplot of observed versus imputed for non-zero observed counts. This serves as a visualization tool to indirectly assess whether the imputed values are sensible for the observed zeros by looking at the performance for observed non-zeros. The imputed values for the non-zero observed counts should not deviate wildly from the observed values even though some level of “smoothing” is being applied.

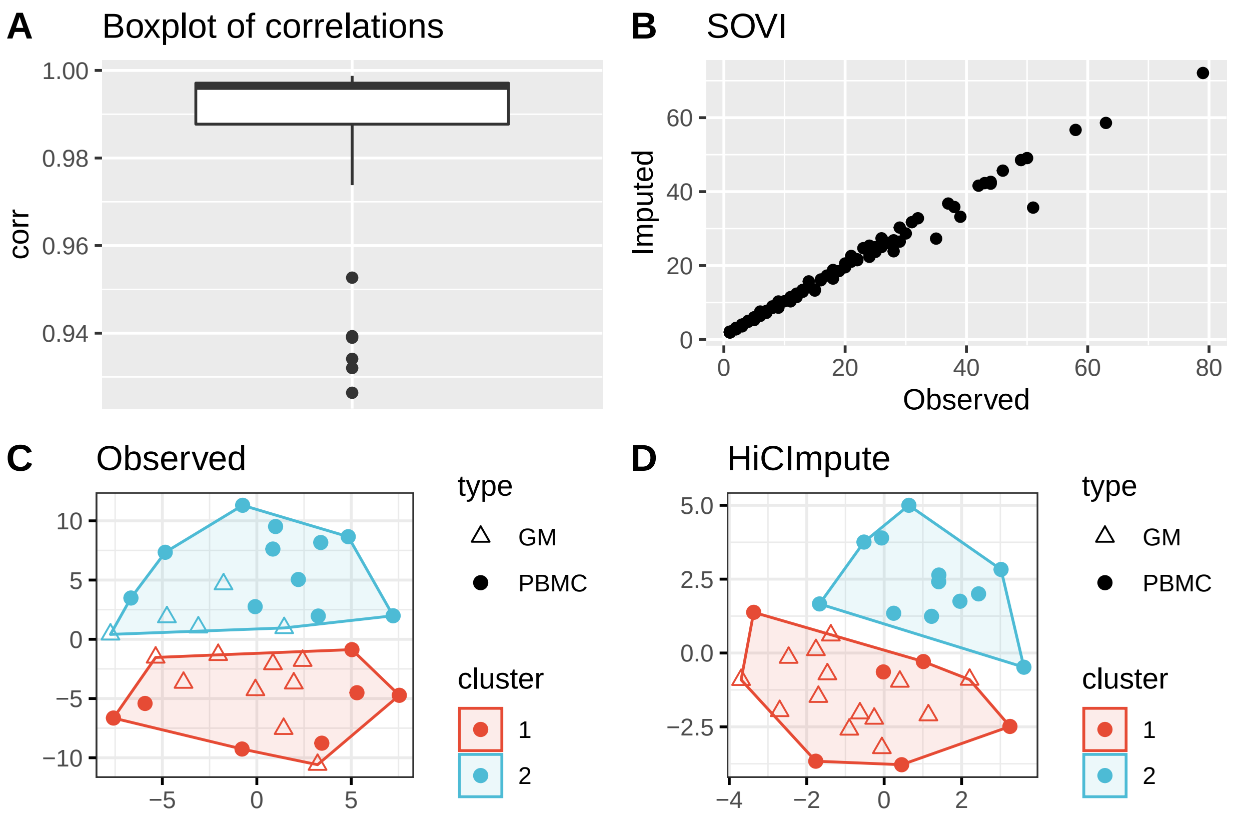
- *scHiC_Kmeans* perform kmeans cluserting analysis on observed and imputed scHi-C matrix. The output is the best clustering result that has the smallest within-cluster sum of squares out of 50 results with 50 starting points each with 200 iterations. One can change the random seed (seed), the number of starting points (nstart) and the maximum number of iterations (iter.max) in scHiC_assess function.
- *scHiC_tSNE* visualizes scHi-C matrix using t-SNE (t-distributed stochastic neighbor embedding) (Van der Maaten et al. 2008) followed by applying Kmeans clustering (Xie et al. 2021).

We use the first scHi-C dataset in Xie et al., 30 loci on chromosome1 of 32 cells (14 GM cells and 18 PBMC cells) from GSE117874, as an example of a real data analysis. Since we know the type of each cell, we imputed two types of cells separately and then combine the imputed values, and call it GSE117874_imp. The first 14 single-cells of GSE117874_imp are GM cells and the remaining 18 single-cells are PBMC cells. The cell_type indicates the underlying cell type.

```
data("GSE117874_chr1_wo_diag")
data("GSE117874_imp")
GSE117874_res=scHiC_assess(scHiC = GSE117874_chr1_wo_diag, imputed = GSE117874_imp,
                           cell_type = c(rep("GM",14),rep("PBMC",18)))
```

The first element of **scHiC_assess** output contains four plots. **A** is the boxplot of correlations between the imputed and the observed on nonzero observations. For the GSE117874 chr1 dataset, we can see that all the correlations are greater than 0.92. **B** is the scatterplot of imputed versus observed on nonzero observations of the first cell, where all the points are densely distributed along the diagonal line. **C** and **D** are the t-SNE (t-distributed stochastic neighbor embedding) visualization of 32 GSE117874 cells, with k-means clustering results based on t-SNE data. We can see that the HiCImpute corrected some of the misclassified cells.

```
GSE117874_res$plots
```



The second and third elements of **scHiC_assess** output are the Kmeans clustering of the observed and imputed scHi-C data, respectively. The following two clustering results shows that, using the default parameter settings, 4 of GM cell and 5 of PBMC cells are misclassified. The HiCImpute-imputed data corrected 3 of

GM misclassified cels and 1 of PBMC misclassified cells are corrected. These results are not expected to be the same as dimension reduction first using t-SNE and then clustering (C and D), but they are quite similar.

```
GSE117874_res$obs_cluster
#>    
#>     GM PBMC
#>   1  4   13
#>   2 10    5
```

```
GSE117874_res$imp_cluster
#>    
#>     GM PBMC
#>   1  1   14
#>   2 13    4
```

3. Funcitons for generating scHi-C data

`scHiC_simulate(data, alpha_0,alpha_1,beta_l=0.9,beta_g=0.9,beta_m=0.9, alpha, n_single)`

data is a matrix of 3D coordinates with each row being the 3D coordinate of a loci.

alpha_0 and **alpha_1** are the parameters that control sequencing depth of data.

beta_l, **beta_g**, and **beta_m** are the parameters that control effect size of covariates.

gamma is the quantile that is used to set the threshold.

eta is the percent of structural zeros that are set to be common structural zeros among all single-cells.

n_single is the number of single cells to be generated.

This function is designed to simulate scHi-C data based on a 3D chromatin structure. It requires 3D coordinates as shown in below. The data coord3D is generated from another package called SIMBA.

```
data("coord3D")
head(coord3D)
#>     [,1]  [,2]  [,3]
#> V1 -0.72  0.85 -1.25
#> V2 -0.69  0.59 -0.51
#> V3 -0.64 -0.40 -1.15
#> V4 -1.07 -0.75 -0.73
#> V5 -0.73 -0.72 -0.78
#> V6  0.39 -0.38 -1.25
```

The following function generates 100 single-cells based on coord3D. The output contains the underline truecount, the position of SZ, and the generated single-cells. Truecounts can be used to measure imputation accuracy.

```
set.seed(1234)
#Generate 100 random type1 single-cells
data <- scHiC_simulate(data=coord3D, alpha_0=5.6,alpha_1=-1,
                        beta_l=0.9,beta_g=0.9,beta_m=0.9, gamma=0.1, eta=0.8, n_single=100)
```