scHiCSRS: An R package implementing "a self-representation smoothing method with Gaussian mixture model for identifying structural zero and enhancing single-cell Hi-C data"

Qing Xie, Shili Lin

8/31/2021

1.Introduction

Single-cell HiC techniques enable us to study the between-cell variability in long-distance interactions and genomic features. However, single-cell HiC data usually suffers from excessive zeroes due to a lack of sequencing depth. Among those zeros in scHiC contact matrix, some are structural zero (SZ) because the corresponding pairs do not interact with each other at all, while others are dropout (DO) as a result of low sequencing depth. While those dropouts happen at random, those structural zeros do not.

scHiCSRS (Xie and Lin, 2021) imputes sparse single-cell Hi-C matrix through a self-representation smoothing approach and distinguishes DO from SZ through a Gaussian mixture model. Different from literature that treats each single-cell separately, we borrow informaton not only from the neighborhood in the same cell, but also from other cells at the same position.

In this package, we provide the following main functions:

- **SRS**: the flagship function of scHiCSRS, which imputes single-cell Hi-C data through a self-representation smoothing approach. The outputs can be used to facilitate downstream analysis of single-cell Hi-C data.
- **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 as a matrix containing the 3D coordinates.

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

2. SRS

SRS(X_count, windowsize=2, nbins, lambda1 = NULL, lambda2 = 1e10, initA = NULL, initB = NULL,ncores = 1, MAX_ITER = 4, ABSTOL = 1e-3, learning_rate = 0.0001, epochs = 100, batch_size = 128, run_batch = TRUE, verbose = TRUE, estimates.only = FALSE)

SRS imputes single-cell Hi-C data through a self-representation smoothing approach. The following show all the parameters in **SRS** function:

2.1 Input data 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. The other two types of formats are either 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 $n \times n$. scHiCSRS automatically transforms these two types of input into a matrix with each column being the vector of upper triangular matrix of a single-cell (i.e., the preferred format). 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 the 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_4k is of dimension 61×61, each single-cell has a vector of length $61 \times 60/2 = 1830$.

```
options(digits = 2)
library(scHiCSRS)
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 the other 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 a length of 100 and each list element is a 61×61 single-cell matrix. The second is a $61 \times 61 \times 100$ array that has 100 matrices of dimension 61×61.

```
data("K562_T1_4k_list")
#showing the 2nd element in the list
K562_T1_4k_list[[2]][1:10,1:10]
#>          locus_1 locus_2 locus_3 locus_4 locus_5 locus_6 locus_7 locus_8
#> locus_1        0       3       2       0       1       0       2       1
#> locus_2        3       0       3       1       4       2       1       2
#> locus_3        2       3       0       4       6       2       2       1
#> locus_4        0       1       4       0       8       1       1       1
#> locus_5        1       4       6       8       0       3       2       1
#> locus_6        0       2       2       1       3       0       6       3
#> locus_7        2       1       2       1       2       6       0       5
#> locus_8        1       2       1       1       1       3       5       0
#> locus_9        5       2       5       1       4       4       8       8
#> locus_10       3       2       1       2       1       4       4       7
#>          locus_9 locus_10
#> locus_1        5        3
#> locus_2        2        2
#> locus_3        5        1
#> locus_4        1        2
#> locus_5        4        1
#> locus_6        4        4
#> locus_7        8        4
#> locus_8        8        7
#> locus_9        0        5
#> locus_10       5        0
```

```
data("K562_T1_4k_3D_array")
#showing the 2nd 2D matrix in the 3D array; should match the above partial list.
```

```
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
```

windowsize is the size of neighborhood region. A windowsize of w results in a (2w+1)*(2w+1) neighboring submatirx.

nbins is the number of bins of the observed single-cell Hi-C matrix.

lambda1 is the tuning parameter to facilitate feature selection and regularization.

lambda2 is the tuning parameter to penalize teh diagonal element of the parameter to eliminate the trivial solution of representing an expression level as a linear combination of itself.

initA is the initialization of A. The elements of A represent the similarities between loci in the same cell.

initB is the initialization of B. The elements of B represent the similarities between all single-cells at the same position.

ncores is the number of cores to use. Default is 1.

MAX_ITER is the Maximum iteration of the external circulation of SRS.

ABSTOL is the absolute tolerance of the external circulation.

learning_rate is a hyper parameter that controls the speed of adjusting the weights of the network with respect to the loss gradient.

epochs is the number of the entire training set going through the entire network.

batch_size is the number of examples that are fed to the algorithm at a time.

run_batch indicates whether to use batch or to set the number of all the samples as teh value of the batch size. Default is TRUE.

verbose indicates whether to output the value of metrics at the end of each epoch. Default is TRUE.

estimates.only indicates whether only estimate or not. If TRUE, than output the SRS imputed matrix. If FALSE, A list of information is outputted.

2.2 Use of SRS function

Here are examples for the use of SRS on the simulated data in three different input format. Note that these examples are for illustration purpose only. For accurate result, users have to run with larger number of epochs.

```
data("K562_T1_4k")
data("K562_T1_4k_true")
set.seed(1234)
T1_4k_result=SRS(scHiC=K562_T1_4k, expected=K562_T1_4k_true,
                 windowsize=2, nbins=61, epochs = 100)
```

```r
data("K562_T1_4k_list")
set.seed(1234)
T1_4k_result2=SRS(scHiC=K562_T1_4k_list, expected=K562_T1_4k_true,
                  windowsize=2, nbins=61, epochs = 5)
```

```r
data("K562_T1_4k_3D_array")
set.seed(1234)
T1_4k_result3=SRS(scHiC=K562_T1_4k_3D_array, expected=K562_T1_4k_true,
                  windowsize=2, nbins=61, epochs = 5)
```

The output of SRS is a list of estimated matrix A and S, the total running time, the observed data, SZ probabilities, the imputed data before zeroing out the SZs (Impute_All), the imputed data zeroing out the SZ entries (Impute_SZ), and the expected or unerlying true data (NULL for real data). The following example is the result of K562_T1_4k data.

```r
data("T1_4k_result")
T1_4k_result$SZ[1:5,1:5]
#>      [,1]    [,2] [,3]    [,4] [,5]
#> [1,]    0 0.0e+00    0 0.0e+00    0
#> [2,]    0 0.0e+00    0 0.0e+00    0
#> [3,]    0 0.0e+00    0 0.0e+00    0
#> [4,]    0 5.6e-14    0 4.3e-14    0
#> [5,]    0 0.0e+00    0 0.0e+00    0
```

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

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

```r
dim(T1_4k_result$A)
#> [1] 1830 1830
```

```r
dim(T1_4k_result$S)
#> [1] 100 100
```

```r
T1_4k_result$total.time
#> Time difference of 2.5 mins
```

2.3 Assessing scHiCSRS imputation results

`scHiC_assess(result, cell_index, n, cell_type, dims = 2, perplexity = 10, seed = 1000, kmeans = TRUE, ncenters = 2)`

result is a list of observed, expected, and imputed values (SRS output) for simulated data or a list of observed and imputed values for the real dataset.

cell_index indicates which single-cell is used to generate an example 2D heatmaps and scatterplot of observed versus imputed or expected versus imputed. 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 (expected = NULL for real data) of the functions. We illustrates the usage of these two cases separately.

2.3.1 Assessing result for 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 proportion of true structural zeros (PTSZ) and proportion of true dropouts (PTDO) when setting different thresholds for calling an observed zero to be an structural zero (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 correctly identified as such by a method.
- *PTDO*: Proportion of true dropouts. This is defined as the proportion of underlying sampling zeros (due to insufficient sequencing depths) correctly identified as such by a method.
- *MCA*: Mean correlation between the imputed and expected counts for all observed values.
- *MAEZ*: Mean 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.
- *MAEA*: Mean 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.

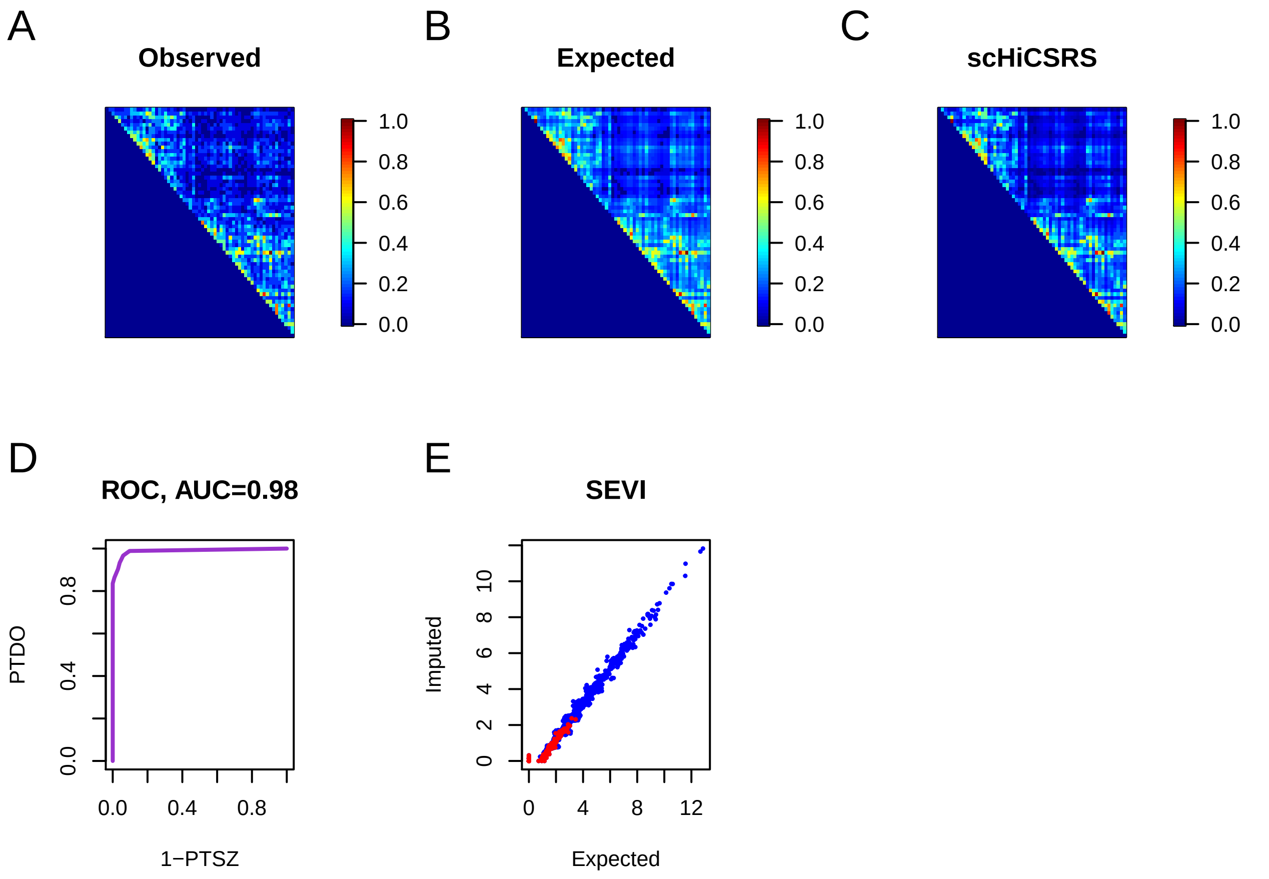
For a preset PTSZ level, scHiC_assess also calculates PTDO. Default is seet to be 0.95, but the user may set it to a different value.

The following is an example output of *scHiC_assess*.

The numerical summary includes the mean and standard deviation of the 6 measurements, the PTDO when PTSZ is fixed to be 0.95, and plots as discussed. **A-C** are the heatmaps of observed, expected, and imputed single-cell for the first cell of K562_T1_4k, where we can see that scHiCSRS is able to denoise and recover the underlying structure. **D** is the ROC curve of the K562_T1_4k. **E** is the scatterplot of the imputed versus expected values for the first K562_T1_4k cell, with the red points being the observed zeros.

```
scHiC_assess(result = T1_4k_result, n=61)
#> $summary_mean
#>   PTSZ PTDO MAEZ MAEA  MCA
#> 1 0.96 0.92 0.59 0.76 0.93
#>
#> $summary_se
#>     PTSZ   PTDO   MAEZ    MAEA     MCA
#> 1 0.0015 0.0013 0.0026 0.00056 0.00032
#>
```

```
#> $PTSZ_95
#>   PTDO     SD2 thresh
#> 1 0.95 0.0095   0.98
```



2.3.2 Assessing result for 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.

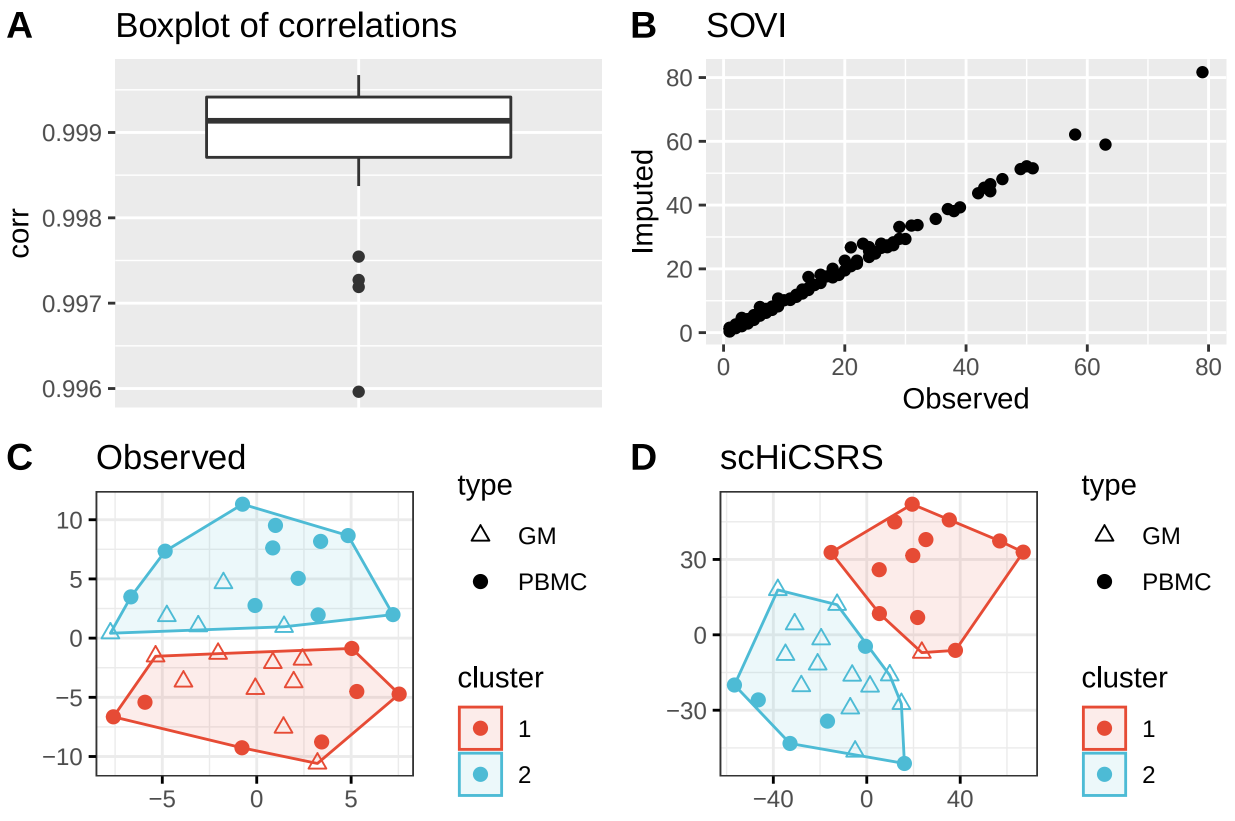
- *Boxplot* illustrates boxplot of correlations between imputed and observed values on nonzero observations.
- *SOVI* draws scatterplot of observed versus imputed for non-zero observed counts. This serves as a visualization tool to indirectly assess whether the imputed values are sensible for the observed 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* performs kmeans clusterting 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 multiple results with different starting points. 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 (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 and 18 PBMC - 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. Note that users have to create a list of observed, imputed for real data as an input.

```
data("GSE117874_chr1_wo_diag")
data("GSE117874_imp")
options(digits = 6)
#scHiC takes the three types as described in SRS.
GSE117874_result=list(scHiC=GSE117874_chr1_wo_diag, Imputed=GSE117874_imp, Expected=NULL)
GSE117874_summary=scHiC_assess(result = GSE117874_result,
                                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 the 32 GSE117874 cells, with k-means clustering results based on the t-SNE data. We can see that the scHiCSRS corrected some of the misclassified cells.

```
GSE117874_summary$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, 5 of GM cell and 7 of PBMC cells are misclassified. The scHiCSRS-imputed data corrected 3 of the GM misclassified cells. These results are not expected to be the same as performing dimension reduction first using t-SNE and then clustering (C and D), but they are quite similar.

```
GSE117874_summary$obs_cluster
#>
#>      GM PBMC
#>    1  4    13
#>    2 10     5
```

```
GSE117874_summary$imp_cluster
#>
#>      GM PBMC
#>    1  1    13
#>    2 13     5
```

3. scHi-C data generating function

This function is designed to simulate scHi-C data based on a 3D chromatin structure.

`scHiC_simulate(data, alpha_0,alpha_1, alpha, gamma, eta, n_single)`

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

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. Their default values are set to be 0.9 as in paper.

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.

We provide the 3D coordinates of the three K562 single-cells that were used in Xie et al. 2021. The following is an example of the 3D coordinates of the first single-cell.

```
data("coord3D_T1")
head(coord3D_T1)
#>          [,1]      [,2]      [,3]
#> V1 -0.718904  0.854780 -1.249469
#> V2 -0.687047  0.589028 -0.513380
#> V3 -0.643530 -0.401648 -1.152007
#> V4 -1.068727 -0.753920 -0.730272
#> V5 -0.734480 -0.718075 -0.777482
#> V6  0.393062 -0.377485 -1.251307
```

The simulation procedure is based on the function $\log(\lambda_{ij}) = \alpha_0 + \alpha_1 \log d_{ij} + \beta_l \log(x_{l,i}x_{l,j}) + \beta_g \log(x_{g,i}x_{g,j}) + \beta_m log(x_{m,i}x_{m,j})$ followed by Park et al. (2019). d_{ij} is the 3D distance of pair (i,j), α_1 is set to -1 to follow the typical biophysical model, α_0 is the scale parameter. On the other hand, $x_{l,i}$, $x_{g,i}$, and $x_{m,i}$ are covariates generated from uniform distributions to mimic fragment length, GC content, and mappability score, respectively, and their coefficients, the β's, are all set to be 0.9.

We find the lower $\gamma\%$ quantile of the λ_{ij} as the threshold, for those $\lambda_{ij} <$ threshold, randomly select half of them to be candidates for structural zeros. Among these candidates, randomly select $\eta\%$ of them and set their new λ_{ij} value to be zero. These are the SZs across all SCs.

The following function generates 100 single-cells based on coord3D_T1. The output contains the underlying expected counts, the position of SZ, and the generated single-cells 2D matrices in the preferred format.

```
set.seed(1234)
#Generate 100 random type1 single-cells
T1_simulate <- scHiC_simulate(data=coord3D_T1, 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)
```

The following is an example of running **SRS** on the simulated data.

```
scHiC=T1_simulate$singledat
set.seed(1234)
T1_simulate_result=SRS(scHiC=T1_simulate$singledat, expected=T1_simulate$truecount, windowsize=2, nbins=
```

Reference

- Xie, Q, & Lin, S. (2021). scHiCSRS: A Self-Representation Smoothing Method with Gaussian Mixture Model for Identifying Structural Zeros and Enhancing single-cell Hi-C Data.

- Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(11).

- Park, J. and Lin, S. (2019) Evaluation and comparison of methods for recapitulation of 3d spatial chromatin structures. Briefings in bioinformatics, 20(4):1205–1214.