scHiCSRS: Imputation of Single Cell HiC Data Using A Self Representation Smoothing Approach

Qing Xie, Shili Lin

6/27/2021

```r
library(scHiCSRS)
library(mclust)
#> Package 'mclust' version 5.4.7
#> Type 'citation("mclust")' for citing this R package in publications.
```

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 SRS, which imputes single cell HiC data through a self-representation smoothing approach. The outputs can be used to facilitate downstream analysis of single cell HiC data.
- **kmeans_cluster**: perform kmeans cluserting analysis on observed or imputed scHiC matrix.
- **tsne_plot**: visualize scHiC matrix with through tsne (t-distributed stochastic neighbor embedding) procedure that makes it easier to visualize the cell clusters. It can further apply kmeans clustering on tsne data.
- **generate_single**: simulates single cell HiC data based on 3D coordinates of chromosome.
- **heatmap**: visualize the data as heatmap.
- **summa**: summarizes imputation accuracy of simulation study.
- **summa2**: calculates PTDO (proportion of true dropouts) when PTSZ (proportion of structural zero) is fixed to be 0.95.
- **summa3**: calculates PTSZ when PTDO is fixed to be 0.80.
- **SRS_ROC**: draws ROC curve of imputed data, which can be used to compare diagonostic ability.

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

2. SRS

SRS imputes single cell HiC data through a self-representation smoothing approach.

2.1 Input data format

The following show all the parameters in **SRS** function:

SRS <- function(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)

X_count is the observed matrix, with each column being the uppertriangular of a single cell HiC matrix. 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 HiC matrix are symmetrical.

Here is an example for the format of single cell matrix:

```
options(digits = 2)
library(SRS)
#>
#> Attaching package: 'SRS'
#> The following objects are masked from 'package:scHiCSRS':
#>
#>     calc.lambda, calc.size.factor, clean.data, dmix, generate_single,
#>     get_mix, GM_Psz, hm, keras_lasso_regression, kmeans_cluster, lnorm,
#>     log_normalization, loss_define, objective, regularizer_define, SRS,
#>     SRS_ROC, summa, summa2, summa3, tsne_plot
data(simudat)
head(simudat)
#>      V1 V2 V3 V4 V5 V6 V7 V8 V9 V10
#> [1,]  7  7  7  4  3  6  6  1  5   5
#> [2,]  1  4  2  1  2  3  1  5  2   5
#> [3,]  4  3  1  4  3  6  1  1  4   3
#> [4,]  1  1  0  2  1  5  1  2  3   2
#> [5,]  1  4  0  3  1  8  2  4  4   3
#> [6,]  9  4  9  3  7  4  6  4  3   7
```

In this example, there are 10 single cells, and each column is a vector of the upper triangular of each single cell. Since this simudat is in dimension 61×61 so that each single cell has a vector of length $61 \times 60/2 = 1830$.

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 HiC 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 Numerical summary of SRS results

Here is a example for the use of SRS:

```
data("simudat")
#simudat_res=SRS(simudat, windowsize=2, nbins=61, epochs = 100, estimates.only = FALSE)
```

The output of SRS is a list of imputed matrix, standard error of estimates, information about dataset, size factor used for normalization, time taken to generate predictions, time taken to compute the posterior distribution, and total time for SRS estimation. The following is the imputed matrix.

```
data("simudat_res")
head(simudat_res)
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,]  5.8  5.8  5.8  5.4  5.3  5.5  6.0  5.2  5.4   5.4
#> [2,]  2.7  3.1  2.6  2.6  2.7  3.0  2.6  2.9  2.7   2.8
#> [3,]  3.9  4.2  3.9  4.3  4.0  4.2  3.8  3.9  4.1   4.2
#> [4,]  1.8  1.8  1.6  1.9  1.8  2.6  1.9  2.1  2.3   2.0
#> [5,]  3.2  3.9  3.1  3.5  3.5  4.1  3.6  3.7  3.6   3.5
#> [6,]  6.7  5.8  6.2  6.0  6.2  6.0  6.3  6.1  5.8   6.1
```

2.3 Psz calculation

To make inference on SZ and DO, we apply a Gaussian mixture model on imputed data at each position separately and calculata Psz definded as the proportion of Gaussian component with lower mean. Here is a function to calculate Psz for a given observed and its imputed matrix.

```
  Psz=GM_Psz(simudat, simudat_res, nmix=4)
  head(Psz)
#>      [,1] [,2]     [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,]    0    0  0.0e+00    0    0    0    0    0    0     0
#> [2,]    0    0  0.0e+00    0    0    0    0    0    0     0
#> [3,]    0    0  0.0e+00    0    0    0    0    0    0     0
#> [4,]    0    0  1.8e-53    0    0    0    0    0    0     0
#> [5,]    0    0 3.8e-286    0    0    0    0    0    0     0
#> [6,]    0    0  0.0e+00    0    0    0    0    0    0     0
```

2.4 Kmeans clustering

scHiCSRS provides kmeans function to cluster observed or imputed scHiC data. The following shows that the 10 imputed cells are clustered into two clusters of size 4 and 6 respectively.

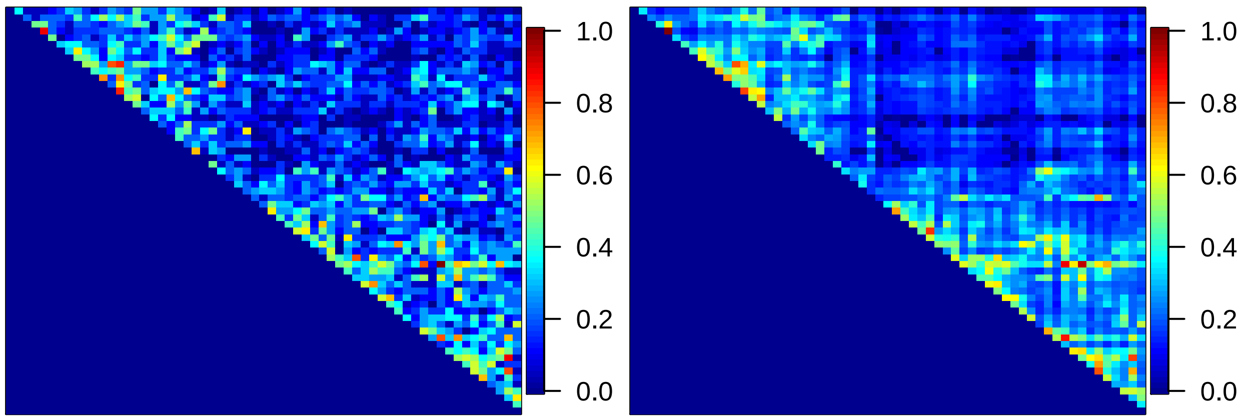
```
cluster=kmeans_cluster(simudat_res, centers=2, nstart=20, iter.max=200, seed=1250)
table(cluster$cluster)
```

```
#>
#> 1 2
#> 4 6
```

2.5 Heatmap

hm draws heatmap of HiC data so that we can visually compares the imputation results. For example, the following is the heatmap of observed and imputed single cell of the simudat, where we can see an improvement of sequence depth.

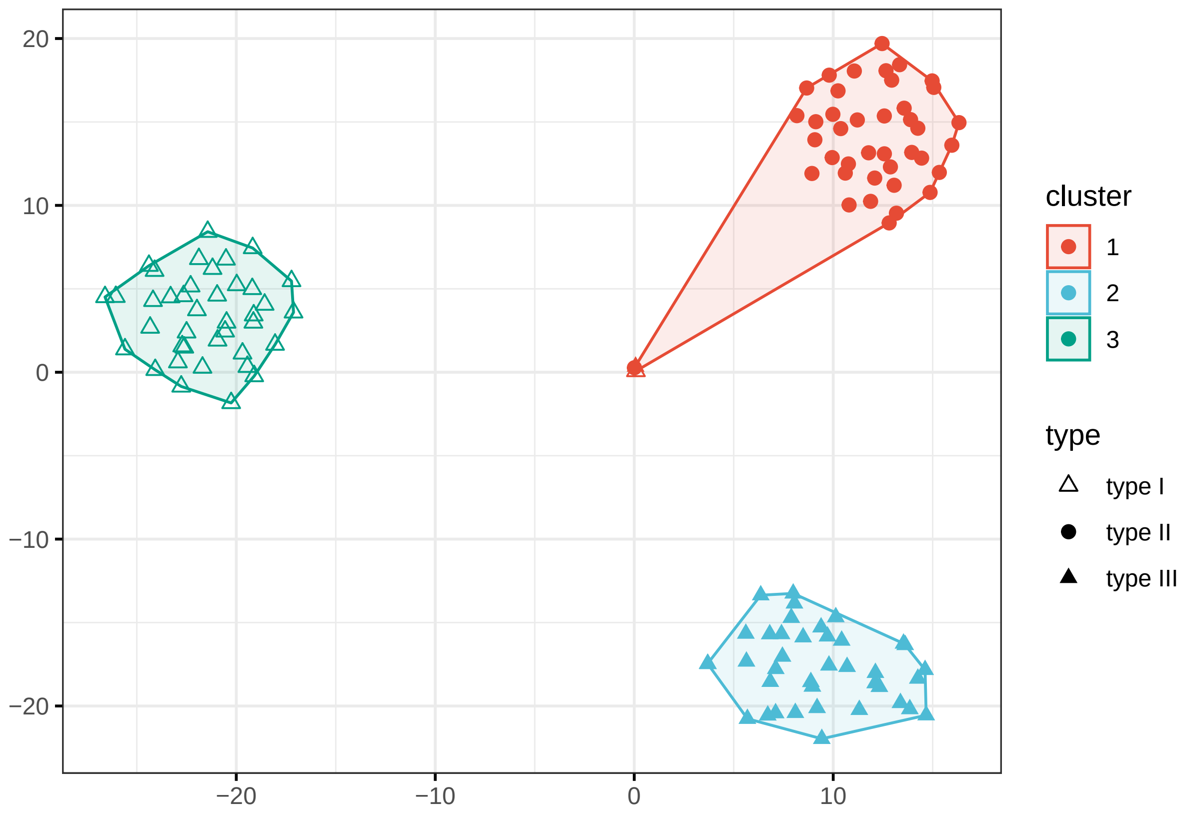
```
par(mar = c(0.4,0.4,0.4,0.4))
par(mfrow=c(2,2))
hm(simudat[,1], 61)
hm(simudat_res[,1], 61)
```



2.6 tsne visualization

scHiCSRS can reduce dimension of scHiC data through tsne (t-distributed stochastic neighbor embedding) visualizaiton that makes it easier to visualize the cell clusters. The following example is a tsne visualization of 3 types of imputed data, with kmeans clusering results based on tsne data. We can see that most of cells are correctly clustered with only one typeI cell being mistakenly clustered into typeII cluster.

```
library(Rtsne)
library(ggpubr)
#> Loading required package: ggplot2
data("simba_imp")
tsne_plot(simba_imp, cell_type=c(rep("type I",40),rep("type II",40),rep("type III",40)), dims = 2, perpl
```



3. Funcitons for generating scHiC data

3.1 Generate single cell

generate_single is a function designed to simulate scHiC data based on 3D structure of chromosome. It requires 3D coordinates as shown in below. The data str1 is an example of 3D coordinates of 61 segments with each row being the coordinate of a segment.

```
data("str1")
head(str1)
#>     [,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
```

And the idea of simulation is based on the function $log(\lambda_{ij}) = \alpha_0 + \alpha_1 log(d_{ij}) + \beta_l log(x_{g,i}x_{g,j}) + \beta_m log(x_{x_{m,i},x_{m,j}})$, where α_0, α_1 are set to control the sequence depth, and $x_{l,i} \sim Unif(0.2, 0.3)$, $x_{g,i} \sim Unif(0.4, 0.5)$, $x_{m,i} \sim Unif(0.9, 1)$ are used to account for covariates.

The following function generates 10 single cells based on str1. 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 <- generate_single(data=str1, alpha_0=5.6,alpha_1=-1, beta_l=0.9,beta_g=0.9,beta_m=0.9, alpha=0.2,
```

3.2 Accuracy summary

summa summarizes imputation accuracy using the 11 measurements used in the paper.

```
data("simudat_true")
options(digits = 2)
summa(simudat_true, simudat_res,simudat, Psz)
#> $mean
#>   PTSZ PTDO  MSE RE_sampling RE_all AE_0 AE_sampling AE_all cor_0 cor_sampling
#> 1    1 0.97 0.12        0.16  0.077  0.2        0.35   0.25  0.98         0.93
#>   cor_all
#> 1    0.82
#>
#> $sd
#>     PTSZ  PTDO   MSE RE_sampling RE_all AE_0 AE_sampling AE_all  cor_0
#> 1 0.0059 0.013 0.013        0.16  0.077 0.27        0.27   0.23 0.0053
#>   cor_sampling cor_all
#> 1        0.017  0.0054
```

4 Reproduce simulation study results

In my simulation study, I generated three types of single cells based on three existing single cell contact data. For each type of single cells, I considered three sequencing depth (2k, 4k, and 7k) and three levels of cell numbers (10, 50, 100), resulting in 27 cases. The simulated data are also available in this package. In order to reproduce the simulation study result, you can run the following commented codes.

```
# data(simba1)
# data(simba1_true)
# data(simba2)
# data(simba2_true)
# data(simba3)
# data(simba3_true)
# data(simba4)
# data(simba4_true)
# data(simba5)
# data(simba5_true)
# data(simba6)
# data(simba6_true)
# data(simba7)
# data(simba7_true)
# data(simba8)
# data(simba8_true)
# data(simba9)
# data(simba9_true)
#
# observed=list(simba1, simba2, simba3, simba4, simba5, simba6, simba7, simba8,simba9)
# true=list(simba1_true, simba2_true, simba3_true, simba4_true, simba5_true, simba6_true, simba7_true, 

# result=list()
# kkk=1
#
# for(i in 1:9)
# {
#   truefile=true[[i]]
#   observedfile=observed[[i]]
#   for(k in c(100,50,10))
#   {
#     observedfile=as.matrix(observedfile[,1:k])
#     truefile=as.matrix(truefile[,1:k])
#     res=SRS(X_count = observedfile+0.0001, windowsize=2, nbins=61, epochs = 100)
#     res2=res$estimate
#     predictfile=NULL
#     for(j in 1:ncol(res2))
#     {
#       temp=matrix(0,61,61)
#       temp[upper.tri(temp, diag = TRUE)]=res2[,j]
#       predictfile=cbind(predictfile,temp[upper.tri(temp)])
#     }
#     result[[kkk]]=list(truefile,predictfile,observedfile2)
#     kkk=kkk+1
#   }
# }

# scHiCSRS_result=result
#
# res_mean = NULL
# res_sd = NULL
# res_PTDO = NULL
# res_PTSZ = NULL
# for (i in 1:length(scHiCSRS_result)) {
#   res <- do.call(summa, scHiCSRS_result[[i]])
#   res_PTDO <- rbind(res_PTDO, do.call(summa2, scHiCSRS_result[[i]]))
#   res_PTSZ <- rbind(res_PTSZ, do.call(summa3, scHiCSRS_result[[i]]))
#   res_mean =rbind(res_mean, res$mean)
#   res_sd =rbind(res_sd, res$sd)
# }

#scHiCSRS_res_mean <- cbind(cells=rep(c(100,50,10),9), res_mean)
#scHiCSRS_res_sd <- cbind(cells=rep(c(100,50,10),9), res_sd)
#scHiCSRS_res_PTDO <- cbind(cells=rep(c(100,50,10),9), res_PTDO)
#scHiCSRS_res_PTSZ <- cbind(cells=rep(c(100,50,10),9), res_PTSZ)
```