scHiCB: Imputation of Single Cell HiC Data Using Bayesian Hierarchical Model

Qing Xie, Shili Lin

7/10/2021

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.

scHiCB (Xie and Lin, 2021) distinguishs DO from SZ and make imputations at the DO positions, based on a Bsyesian hierarchical model that take information from neighborhood, similar cells, and bulk data.

In this package, we provide the following main functions:

- **MCMCImpute**: the flagship function of scHiCB, which imputes single cell HiC data under a Bayesian framework. The outputs can be used to facilitate downstream analysis of single cell HiC data.
- **generate_single**: simulates single cell HiC data based on 3D coordinates of chromosome.
- **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.
- **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.
- **scHiCB_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. MCMCImpute

MCMCImpute imputes single cell HiC data under a Bayesian framework.

2.1 Input data format

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

MCMCImpute(niter = 30000,burnin = 15000,single,bulk = bulk,startval = c(100, 100, 10, 8, 10, 0.1, 900, 0.2, 0, replicate(dim(single)[2], 8)),n,epsilon1 = 0.5,epsilon2 = 5,mc.cores = 1,cutoff = 0.5)

niter is the number of iterations for MCMC. The default value is 30,000.

burnin is the number of burn-in interation. The default value is 15,000.

single is a single cell 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 HiC matrix are symmetrical. You can use function $mattovec(\cdot)$ to transform your matrix into its upper triangular vector.

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

```
options(digits = 2)
library(scHiCB)
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 $61 \times 60/2 = 1830$.

bulk is a vector of bulk data. Bulk data is a combination of single cells, and it provides information for prior settings. If bulk data is not availble, set it to be NULL, and MCMCImpute will sum up the single cells to construct a bulk data.

startval is the starting value of MCMC chain. The default value is $c(100, 100, 10, 8, 10, 0.1, 900, 0.2, 0, replicate(dim(single)[2], 8$

n is the number of bins in the single cell.

epsilon1 is the range size of δ that is uniformly distributed. The default value is 0.5.

epsilon2 is the range size of B that is uniformly distributed. The default value 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.

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, that pair of bins is treated as SZ.

2.2 Numerical summary of MCMCImpute results

MCMCImpute provides a list of posterior mean of SZ probability, imputed data without defining SZ, and imputed data after defining SZ.

Here is a example for the use of MCMCImpute:

```
data("simba1_7k")
data("bulk")
#T1_7k_res=MCMCImpute(niter=100000,burnin=5000,single=simba1_7k,bulk=bulk,startval=c(100,100,10,8,10,0.
#T1_7k_imp=T1_7k_res[[3]]
```

The output of MCMCImput is a list of posterior mean of probability, the imputed data without defining SZ, and imputed data with SZ, using the threshold. The posterior mean of probability is a matrix of size $n \times n$, where n is the number of bins. It tells us the probability of being a SZ between the corresponding pairs. For example, the probability of bin1 and bin2 do not interact is 0.25. IMP1 and IMP2 have the same dimension as the observed data.

```
data("simudat_res")
simudat_res$pii[1:10,1:10]
#>       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#>  [1,]    0 0.25 0.24 0.30 0.30 0.24 0.30 0.25 0.25  0.26
#>  [2,]    0 0.00 0.25 0.30 0.25 0.30 0.26 0.25 0.25  0.24
#>  [3,]    0 0.00 0.00 0.25 0.25 0.25 0.25 0.25 0.25  0.24
#>  [4,]    0 0.00 0.00 0.00 0.24 0.31 0.31 0.25 0.24  0.30
#>  [5,]    0 0.00 0.00 0.00 0.00 0.25 0.31 0.25 0.25  0.31
#>  [6,]    0 0.00 0.00 0.00 0.00 0.00 0.25 0.30 0.25  0.24
#>  [7,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.24 0.25  0.26
#>  [8,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.26  0.24
#>  [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
```

```
head(simudat_res$IMP1)
#>      imp imp imp imp imp imp imp imp imp imp
#> [1,] 7.6 7.6 7.5 5.6 4.6 7.3 6.9 2.8 6.4 6.4
#> [2,] 2.9 5.7 4.0 2.8 4.1 4.8 3.0 6.4 3.8 6.4
#> [3,] 5.6 5.0 2.8 5.9 5.1 6.9 3.1 2.8 5.5 5.0
#> [4,] 3.0 3.0 8.0 4.0 3.2 6.2 3.2 3.9 5.1 3.9
#> [5,] 3.0 6.0 8.0 4.8 2.9 8.3 4.0 5.7 5.8 4.9
#> [6,] 8.9 5.7 8.9 4.7 7.4 5.5 7.0 5.7 4.8 7.8
```

```
head(simudat_res$IMP2)
#>      imp imp imp imp imp imp imp imp imp imp
#> [1,] 7.6 7.6 7.5 5.6 4.6 7.3 6.9 2.8 6.4 6.4
#> [2,] 2.9 5.7 4.0 2.8 4.1 4.8 3.0 6.4 3.8 6.4
#> [3,] 5.6 5.0 2.8 5.9 5.1 6.9 3.1 2.8 5.5 5.0
#> [4,] 3.0 3.0 8.0 4.0 3.2 6.2 3.2 3.9 5.1 3.9
#> [5,] 3.0 6.0 8.0 4.8 2.9 8.3 4.0 5.7 5.8 4.9
#> [6,] 8.9 5.7 8.9 4.7 7.4 5.5 7.0 5.7 4.8 7.8
```

2.3 Kmeans clustering

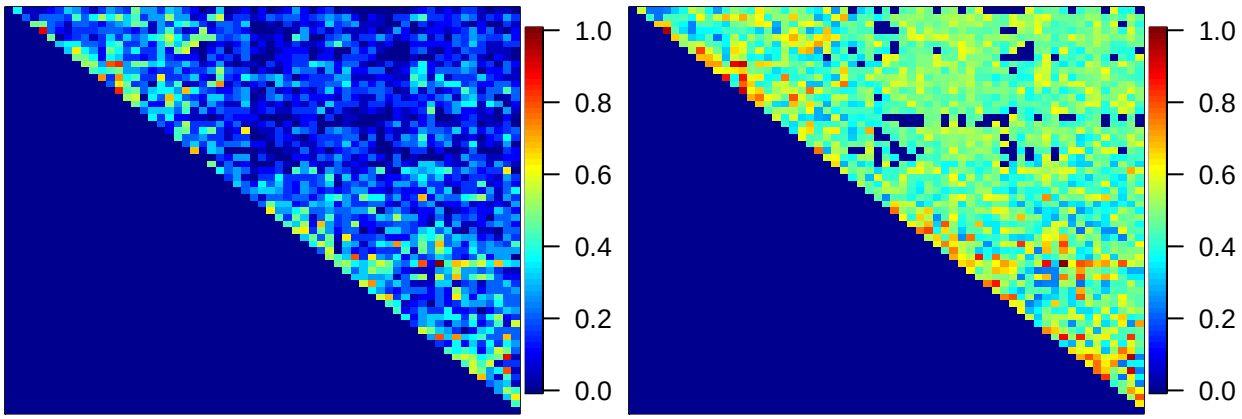
scHiCB provides kmeans function to cluster observed or imputed scHiC data. The following shows that the 10 imputed cells are clustered into three clusters of size 10 in each cluster.

```
data("simba123_imp")
cluster=kmeans_cluster(simba123_imp, centers=3, nstart=20, iter.max=100, seed=1234)
table(cluster$cluster)
#>
#>  1  2  3
#> 10 10 10
```

2.4 Heatmap of the matrix

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.

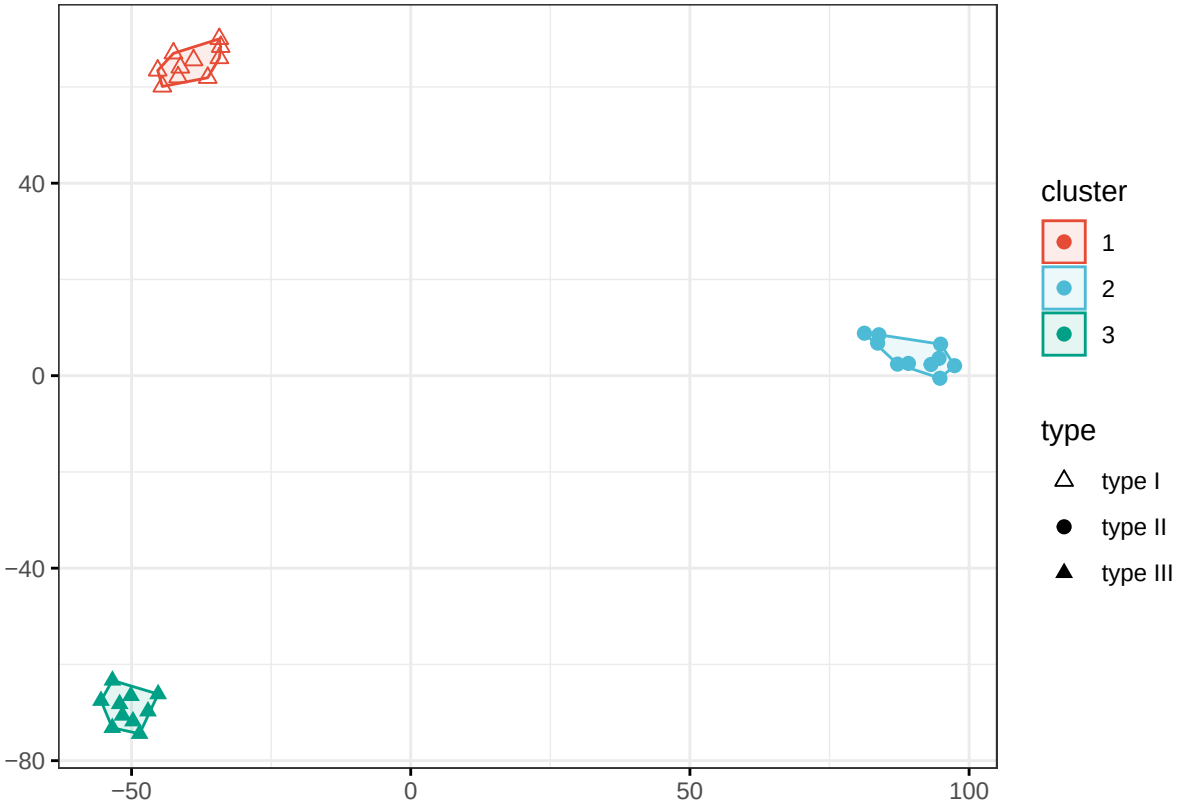
```
par(mar = c(0.4,0.4,0.4,0.4))
par(mfrow=c(2,2))
hm(simudat[,1], 61)
hm(simudat_res$IMP2[,1], 61)
```



2.5 tsne visualization

scHiCB 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 all cells are correctly clustered.

```
library(Rtsne)
library(ggpubr)
#> Loading required package: ggplot2
tsne_plot(simba123_imp, cell_type=c(rep("type I",10),rep("type II",10),rep("type III",10)), dims = 2, p
```



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 generated from another package called SIMBA.

```
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, simudat_true, simudat_res$IMP2)
#> $summa_mean
#>   PTSZ PTDO MSE RE_sampling RE_all AE_0 AE_sampling AE_all cor_0 cor_sampling
#> 1 0.98 0.94   9         2.2      1  2.7         4.6    2.6  0.86         0.21
#>   cor_all
#> 1    0.75
#>
#> $summa_se
#>    PTSZ  PTDO  MSE RE_sampling RE_all AE_0 AE_sampling AE_all cor_0
#> 1 0.013 0.013 0.24           1   0.85  2.5         1.2    1.5 0.018
#>   cor_sampling cor_all
#> 1        0.043  0.0079
```

summa2 calculates PTDO when PTSZ is fixed to be 0.95. **summa3** calculates PTSZ when PTDO is fixed to be 0.80. Their output also contains the standard deviation and the corresponding threshold.

```
summa2(simudat, simudat_true, simudat_res)
#>   PTDO  SD2 thresh
#> 1 0.98 0.01    0.6
```

```
summa3(simudat, simudat_true, simudat_res)
#>   PTSZ SD1 thresh
#> 1    1   0   0.41
```

scHiCB_ROC draws ROC (Receiver operating characteristic) curve to visually demonstrate ability to tell

SZ from DR. The following is an example on the simudat, where we can see the ROC curve goes up to 1 pretty fast.

```
scHiCB_ROC(simudat, simudat_true, simudat_res)
```