scHiCBayes: A Bayesian Hierarchical Model for Identifying Structural Zeros and Enhancing Single Hi-C Data

Qing Xie, Chenggong Han, Victor Jin, Shili Lin

7/28/2021

1.Introduction

Single cell Hi-C techniques enable one to study between-cell variability in long distance inter- actions and genomic features. However, single cell Hi-C (scHi-C) data can suffer severely from sparsity, that is, the existence of excess zeros due to insufficient sequencing depth. Complicate things further is the fact that not all zeros are created equal, as some are due to the loci truly not interacting because of the underlying biological mechanism (structural zeros), whereas others are indeed due to insufficient sequencing depth (sampling zeros), especially for loci that interact infrequently. Differentiating between structural zeros and sampling zeros is important since correct inference would improve downstream analyses such as clustering and discovery of subtypes.

scHiCBayes (Xie, Han, Jin, and Lin, 2021) is a Bayesian hierarchy model that goes beyond data quality improvement by also identifying observed zeros that are in fact structural zeros. scHiCBayes 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**: the flagship function of scHiCBayes, which imputes single cell Hi-C data under a Bayesian framework. The outputs can be used to facilitate downstream analysis of single cell Hi-C data.
- **generate_single**: simulates single cell Hi-C data based on 3D coordinates of chromosome.
- **kmeans_cluster**: perform kmeans cluserting analysis on observed or imputed scHi-C matrix.
- **tsne_plot**: visualize scHi-C 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.
- **hm**: visualize the data as heatmap.
- **performance**: summarizes imputation accuracy of simulation study.
- **PTSZ95**: calculates PTDO (proportion of true dropouts) when PTSZ (proportion of structural zero) is fixed to be 0.95.
- **PTDO80**: calculates PTSZ when PTDO is fixed to be 0.80.
- **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 Hi-C data under a Bayesian framework.

2.1 Input data format

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

niter is the number of iterations for MCMC.

burnin is the number of burn-in interation.

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 Hi-C matrix are symmetrical. You can use function $scHiCBayes :: mattovec(\cdot)$ to transform your matrix into its upper triangular vector.

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

```
options(digits = 2)
library(scHiCBayes)
data("K562_T1_7k")
head(K562_T1_7k)
#>      s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s s
#> [1,] 7 7 7 4 3 6 6 1 5 5 7 7 3 8 5 4 3 7 3 8 5 3 4 5 6 4 5 5 6 4 6 5 8 8 9 7 6
#> [2,] 1 4 2 1 2 3 1 5 2 5 5 2 6 1 4 2 3 3 4 4 5 4 1 3 0 1 6 5 2 3 3 4 3 4 1 4 3 5
#> [3,] 4 3 1 4 3 6 1 1 4 3 1 1 4 7 0 1 6 3 2 2 7 3 8 2 5 6 2 5 3 5 4 5 4 1 6 1 6
#> [4,] 1 1 0 2 1 5 1 2 3 2 0 1 2 4 3 3 3 2 0 2 1 1 5 0 0 1 3 5 2 0 4 3 2 4 0 2 3
#> [5,] 1 4 0 3 1 8 2 4 4 3 8 6 3 0 2 4 3 2 5 5 3 7 2 3 1 3 4 1 4 5 2 3 2 6 1 3 6
#> [6,] 9 4 9 3 7 4 6 4 3 7 8 3 3 6 8 4 5 5 7 7 6 5 5 4 5 6 9 7 8 9 8 5 1 4 6 7 9
#>      s s s s  s s s s s s  s s  s  s s s s s s s s s s  s s s s  s s s  s s s s
#> [1,] 3 5 3 7  9 5 6 7 2 4 10 4 11 11 5 3 5 8 6 3 6 3 2  3 8 3 9  5 3 5 14 5 2 6
#> [2,] 7 2 2 4  1 3 2 4 3 0  5 5  2  4 3 3 4 2 0 3 3 2 2  5 3 3 1  1 2 1  0 1 2 6
#> [3,] 4 2 4 4  3 5 3 4 3 5  5 3  2  3 3 3 5 8 4 7 8 7 8  4 6 3 5  4 5 5  2 5 7 4
#> [4,] 0 2 1 2  0 2 0 0 5 4  3 1  4  1 2 1 1 2 3 3 0 1 1  3 3 4 4  1 2 2  0 4 1 1
#> [5,] 4 7 2 6  5 9 2 5 3 4  6 3  1  0 5 4 4 4 3 6 4 5 2  5 4 2 6  1 1 6  3 5 5 5
#> [6,] 8 1 6 4 11 4 5 5 7 6  4 6  5  2 7 6 7 2 8 3 6 5 5 10 5 6 7 11 4 4  4 8 7 8
#>      s s  s s  s  s s s  s s  s s s s s s  s s  s  s s s s s s s s s s
#> [1,] 4 9  5 6  7  8 5 6  5 4 10 2 4 9 8 4 11 3  5  4 6 4 6 8 3 4 6 6 3
#> [2,] 2 1  6 2  2  3 0 1  2 3  2 4 4 5 2 2  2 2  4  1 1 2 2 0 6 3 4 6 1
#> [3,] 8 3  4 4  5  4 2 4  2 5  1 3 6 3 4 6  7 5  4  7 5 4 8 3 4 2 4 7 3
#> [4,] 0 3  1 1  1  4 3 0  3 2  1 6 5 4 5 2  0 4  2  3 1 0 0 5 5 2 1 1 1
#> [5,] 3 3  2 4  3  2 1 4  2 5  4 4 1 4 6 4  1 1  4  3 4 2 5 5 3 5 4 6 7
#> [6,] 5 3 10 6 10 12 6 4 10 3  5 9 9 9 7 3 11 6 10 11 4 7 5 3 5 8 9 7 8
```

In this example, there are 100 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("K562_T1_7k")
data("K562_bulk")
single=K562_T1_7k
#T1_7k_res=MCMCImpute(niter=100000,burnin=5000,single=K562_T1_7k,bulk=K562_bulk,
#startval=c(100,100,10,8,10,0.1,900,0.2,0,replicate(dim(single)[2],8)),n=61,mc.cores = 1,cutoff=0.5)
```

The output of MCMCImpute 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("T1_7k_res")
T1_7k_imp=T1_7k_res$IMP2
T1_7k_res$pii[1:10,1:10]
#>       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#>  [1,]    0 0.25 0.28 0.34 0.26 0.26 0.30 0.27 0.25  0.27
#>  [2,]    0 0.00 0.25 0.27 0.26 0.26 0.27 0.28 0.26  0.25
#>  [3,]    0 0.00 0.00 0.25 0.25 0.26 0.27 0.28 0.26  0.27
#>  [4,]    0 0.00 0.00 0.00 0.25 0.28 0.30 0.33 0.27  0.31
#>  [5,]    0 0.00 0.00 0.00 0.00 0.26 0.27 0.29 0.26  0.27
#>  [6,]    0 0.00 0.00 0.00 0.00 0.00 0.25 0.27 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_7k_res$IMP1)
#> [1] 1830  100
```

```
dim(T1_7k_res$IMP2)
#> [1] 1830  100
```

2.3 Accuracy summary

performance summarizes imputation accuracy using the following measurements used in the paper.

- **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.

```
data("K562_1_true")
options(digits = 2)
performance(observed=K562_T1_7k, expected=K562_1_true, imputed=T1_7k_imp)
#> $summary_mean
#>   PTSZ PTDO AEOZ AEOA CIEZ CIEA
#> 1    1 0.98  0.2 0.77 0.96 0.94
#>
#> $summary_se
#>   PTSZ  PTDO AEOZ AEOA   CIEZ   CIEA
#> 1    0 0.013  0.3 0.69 0.0073 0.0033
```

PTSZ95 calculates PTDO when PTSZ is fixed to be 0.95.

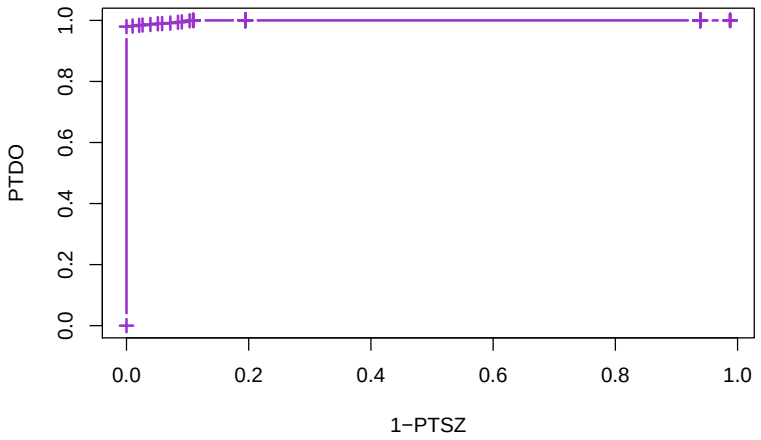
```
PTSZ95(observed=K562_T1_7k, expected=K562_1_true, result=T1_7k_res)
#>   PTDO    SD2 thresh
#> 1 0.99 0.0099   0.57
```

PTDO80 calculates PTSZ when PTDO is fixed to be 0.80. Their output also contains the standard deviation and the corresponding threshold.

```
PTDO80(observed=K562_T1_7k, expected=K562_1_true, result=T1_7k_res)
#>   PTSZ SD1 thresh
#> 1    1   0   0.35
```

ROC draws ROC (Receiver operating characteristic) curve. For a more thorough comparison of methods, it is important to study the interplay between PTSZ and PTDO when setting different thresholds for calling a zero to be an SZ. The following is an example on the K562_T1_7k simulated data, where we can see the ROC curve goes up to 1 pretty fast.

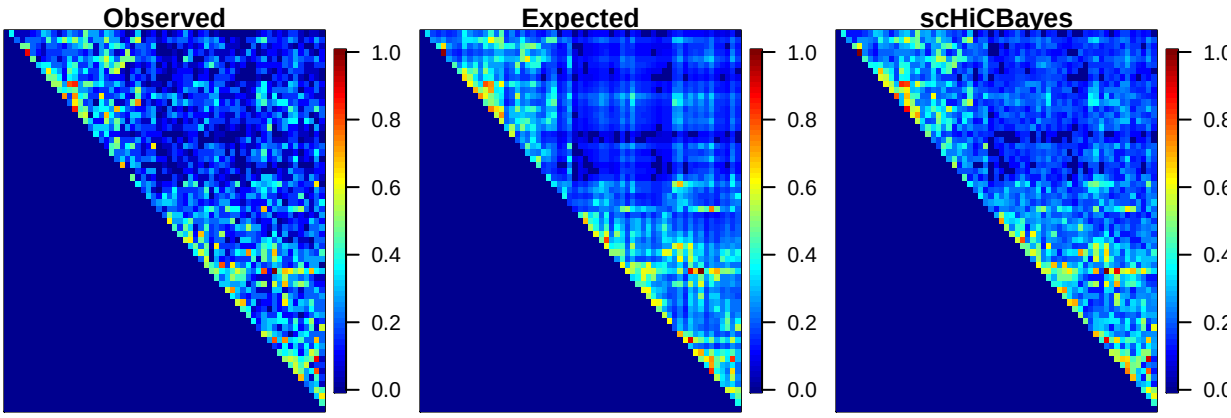
```
ROC(observed=K562_T1_7k, expected=K562_1_true, result=T1_7k_res)
```



2.4 Heatmap of the matrix

hm draws heatmap of Hi-C data so that we can visualize the 2D data matrix before and after the data quality improvement. For example, the following is the heatmap of observed, expected, and imputed single cell of the K562_T1_7k cell1, where we can see that scHiCBayes is able to denoise and recover the underlying structure very well.

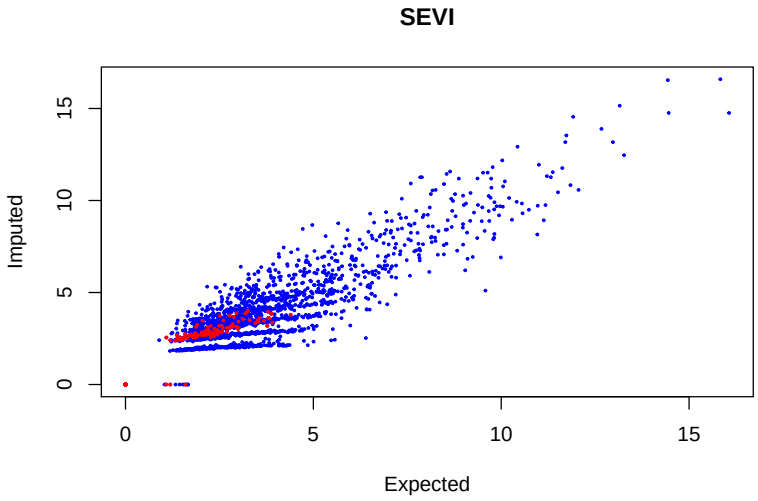
```
data("K562_1_true")
par(mar = c(1,1,1,1))
par(mfrow=c(2,3))
hm(K562_T1_7k[,1], 61, title="Observed")
hm(K562_1_true[,1], 61, title="Expected")
hm(T1_7k_res$IMP2[,1], 61, title="scHiCBayes")
```



2.5 Scatterplot

SEVI draws scatterplot of expected versus imputed. This serves as a visualization tool to directly assess whether dropouts are correctly recovered and accurately imputed.

```
SEVI(observed=K562_T1_7k[,1], expected=K562_1_true[,1], imputed=T1_7k_imp[,1] )
```



3. Funcitons for real scHi-C data

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

```
data("GSE117874_chr1_wo_diag")
# single=GSE117874_chr1_wo_diag[,1:14]
# 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)
```

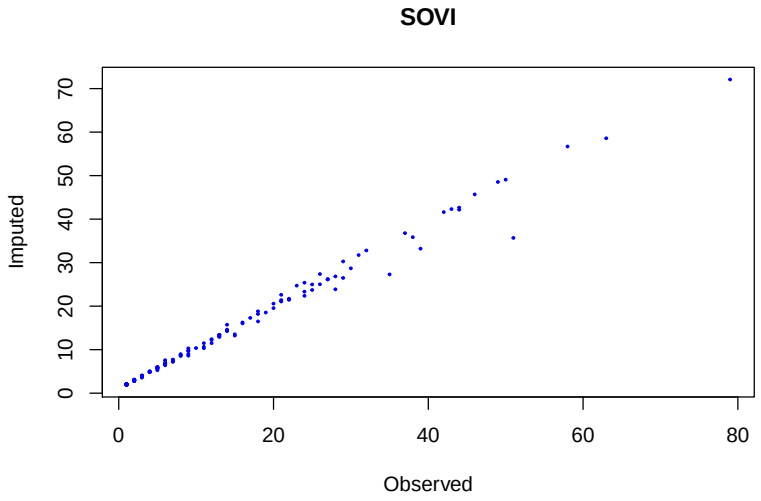
```
#
# single=GSE117874_chr1_wo_diag[,15:32]
# GSE117874_PBMC=MCMCImpute(niter=100000,burnin=5000,single=GSE117874_chr1_wo_diag[,15:32],
#bulk=apply(single,1,sum),startval=c(100,100,10,8,10,0.1,900,0.2,0,replicate(ncol(single),8)),
#n=30,mc.cores = 1,cutoff=0.5)

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

3.1 Scatterplot

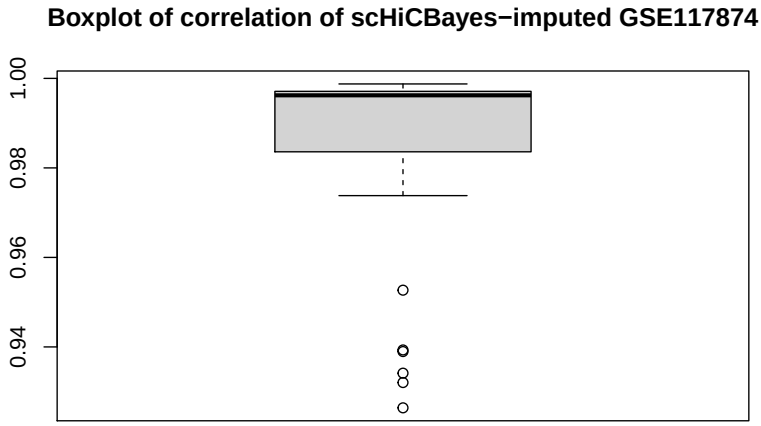
SOVI draws scatterplot of observed versus imputed - applicable to real data application for non-zero observed counts. This serves as a visualization tool to indirectly assess whether the imputed values are sensible for the observed zeros by looking at the performance for observed non-zeros. The imputed values for the non-zero observed counts should not be deviate wildly from the observed values even though some level of "smoothing" is being applied.

```
data("GSE117874_imp")
SOVI(observed=GSE117874_chr1_wo_diag[,1], imputed = GSE117874_imp[,1])
```



We can further calculate the correlation between the observed and imputed values on non-zero observations. The following boxplot is an example of such correlation of 32 GSE117874 cells.

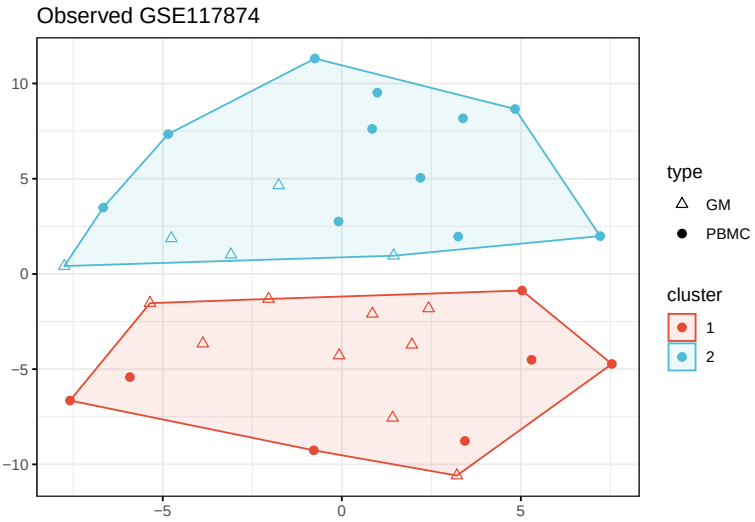
```
corr=NULL
for (i in 1:32) {
  ind=GSE117874_chr1_wo_diag[,i]!=0
  corr[i]=cor(GSE117874_chr1_wo_diag[,i][ind],GSE117874_imp[,i][ind])
}
boxplot(corr, main="Boxplot of correlation of scHiCBayes-imputed GSE117874")
```



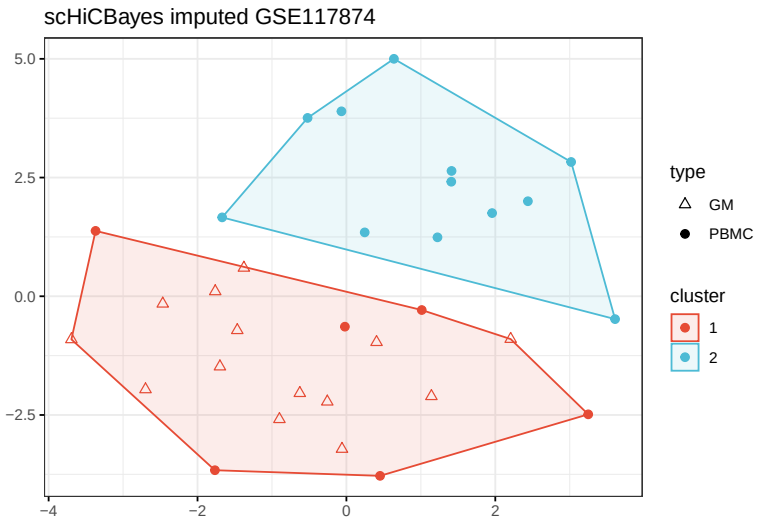
3.2 t-SNE visualization

scHiCBayes can reduce dimension of scHi-C data through t-SNE (t-distributed stochastic neighbor embedding) visualization that makes it easier to visualize the cell clusters. The following example is a t-SNE visualization of 32 GSE117874 cells, with k-means clustering results based on t-SNE data. We can see that the scHiCBayes corrected some of the misclassified cells.

```
tsne_plot(GSE117874_chr1_wo_diag, cell_type=c(rep("GM",14),rep("PBMC",18)),
          dims = 2,perplexity=10, seed=1000, title="Observed GSE117874",
          kmeans = TRUE, ncenters = 2)
```




```
tsne_plot(GSE117874_imp, cell_type=c(rep("GM",14),rep("PBMC",18)),
          dims = 2,perplexity=10, seed=1000, title="scHiCBayes imputed GSE117874",
          kmeans = TRUE, ncenters = 2)
```



3.3 Kmeans clustering

scHiCBayes provides kmeans function to cluster observed or imputed scHi-C data. The following shows that 1 of GM cell and 7 of PBMC cells are misclassified.

```
data("GSE117874_chr1_wo_diag")
data("GSE117874_imp")

cluster=kmeans_cluster(GSE117874_chr1_wo_diag, centers=2, nstart=1, iter.max=1000, seed=1)
sum(cluster$cluster[1:14]==1)
#> [1] 1
sum(cluster$cluster[1:14]==2)
#> [1] 13
sum(cluster$cluster[15:32]==1)
#> [1] 11
sum(cluster$cluster[15:32]==2)
#> [1] 7
```

4. Funcitons for generating scHi-C data

generate_single is a function designed to simulate scHi-C 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 $\log(\lambda_{ij}) = \alpha_0 + \alpha_1 \log d_{ij} + \beta_l \log(x_{l,i}x_{l,j}) + \beta_g \log(x_{g,i}x_{g,j}) + \beta_m log(x_{m,i}x_{m,j})$, 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 such as fragment length, GC content, and mappability score.

The following function generates 100 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, n_single=100)
```