Single-cell ATAC-seq Signal Extraction and Enhancement with SCATE

Zhicheng Ji

Johns Hopkins University,
Baltimore, Maryland, USA
zji4@jhu.edu

Weiqiang Zhou

Johns Hopkins University,
Baltimore, Maryland, USA
wzhou14@jhu.edu

Hongkai Ji

Johns Hopkins University,
Baltimore, Maryland, USA
hji@jhsph.edu

March 26, 2020

Contents

1 Introductions

Single-cell sequencing assay for transposase-accessible chromatin (scATAC-seq) is a new technology for measuring genome-wide regulatory element activities in single cells. With the ability to analyze cells' distinct behaviors in a heterogeneous cell population, this technology is rapidly transforming biomedical research. Data produced by scATAC-seq are highly sparse and discrete. Existing computational methods typically use these data to analyze regulatory pathway activities in single cells. They cannot accurately measure activities of individual cis-regulatory elements (CREs) due to data sparsity. SCATE is a new statistical framework for analyzing scATAC-seq data. SCATE adaptively integrates information from co-activated CREs, similar cells, and publicly available regulome data to substantially increase the accuracy for estimating activities of individual CREs. We show that one can use SCATE to identify cell subpopulations and then accurately reconstruct CRE activities of each subpopulation. The reconstructed signals are accurate even for cell subpopulations consisting of only a few cells, and they significantly improve prediction of transcription factor binding sites. The accurate CRE-level signal reconstruction makes SCATE an unique tool for analyzing regulatory landscape of a heterogeneous cell population using scATAC-seq data.
The main functions of SCATE is demonstrated using the following example of 10 GM12878 and 10 K562 scATAC-seq samples.

2 Data preparation

The first input of SCATE is a list of aligned bam files for scATAC-seq sample. Each single cell should have one separate bam file. One needs to compile a list that includes the location to each bam file. The easist way is to put all bam files in one folder and use `list.files` function to get all locations. Below shows an example where `bamlist` stores the locations to bam files. Alternately users can also use a list of GRanges object as input to SCATE. See section below.

```
# load in SCATE
options(warn=-1)
suppressMessages(library(SCATE))
set.seed(12345)
# set up locations to bam files
bamlist <- list.files(paste0(system.file(package="SCATE"),"/extdata/example"),full.names = T
head(bamlist)
```

```
## [1] "/private/var/folders/xw/5kkj4jvn6y77n1kwtt691w740000gn/T/RtmpkLgxxc/Rinstc2dc6b21dee
## [2] "/private/var/folders/xw/5kkj4jvn6y77n1kwtt691w740000gn/T/RtmpkLgxxc/Rinstc2dc6b21dee
## [3] "/private/var/folders/xw/5kkj4jvn6y77n1kwtt691w740000gn/T/RtmpkLgxxc/Rinstc2dc6b21dee
## [4] "/private/var/folders/xw/5kkj4jvn6y77n1kwtt691w740000gn/T/RtmpkLgxxc/Rinstc2dc6b21dee
## [5] "/private/var/folders/xw/5kkj4jvn6y77n1kwtt691w740000gn/T/RtmpkLgxxc/Rinstc2dc6b21dee
## [6] "/private/var/folders/xw/5kkj4jvn6y77n1kwtt691w740000gn/T/RtmpkLgxxc/Rinstc2dc6b21dee
```

SCATE comes with a built in function to perform cell clustering. Users can also prepare their own clustering results. See Section Cell Clustering.
Users can also rebuild the SCATE database with their own bulk DNase-seq or list of CRE of interest. See Section Build user's own database.
SCATE also supports peak calls instead of bam files as input data format. In this case, the input should be a data frame or list of data frames of scATAC-seq peaks. For each data frame, first column is chromsome name, second column is start site, third column is end site, and fourth column is the number of reads of the peak. Currently `cellcluster` and `SCATE` supports peak calls as input. Please refer to the manual page of `cellcluster` and `SCATE` functions for details.

3 Read in and Preprocessing Data

The function `satacprocess` reads bam files into R as a list of GRanges object. It transforms the reads into the midpoint of the reads (e.g. read chr1:100-150 will be transformed into chr1:125-125). scATAC-seq samples with library size smaller than `libsizefilter` (here is 1000) will be discarded.
Here we run `satacprocess` function using the `bamlist` we prepared in the previous section. The returned `satac` object is a list of Granges with length 18 (2 of the 20 cells are filtered out) and will be used in following analysis.

```
satac <- satacprocess(input=bamlist,type='bam',libsizefilter=1000)
# Number of elements in satac
length(satac)

## [1] 18

# Content in the first element as an example
satac[[1]]

## GRanges object with 10847 ranges and 0 metadata columns:
##           seqnames    ranges strand
##              <Rle> <IRanges>  <Rle>
##       [1]     chr1    569926      +
##       [2]     chr1    569924      +
##       [3]     chr1    569926      +
##       [4]     chr1    569904      -
##       [5]     chr1    569910      -
##       ...      ...       ...    ...
##   [10843]     chrX 153763244      -
##   [10844]     chrX 153775451      +
##   [10845]     chrX 153977428      +
##   [10846]     chrX 154285712      +
##   [10847]     chrX 154842414      +
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

Users can also use a list of GRanges object as input to SCATE by specifying `type='gr'`.
For the purpose of demonstration, we shorten the names of satac.

```
names(satac) <- sub('.*/','',names(satac))
```

4 Cell Clustering (Optional)

SCATE has a built-in function `cellcluster` that clusters cells based on averaged signal of CRE clusters. The below example runs the `cellcluster` function using the `satac` object we generated in previous sections.

```
clusterres <- cellcluster(satac,genome="hg19",clunum=2,perplexity=5)
```

Here `genome` should be either "hg19" or "mm10". It indicates the genome to which the scATAC-seq data is aligned. `clunum` gives the number of clusters. If `clunum=NULL`, the cluster number will be determined automatically by the function. `perplexity` is a number specifying perplexity for tSNE. The default is 30. It should be reduced when sample size is small. For example here it is reduced to 5.
The output of the function `cluster` has three components: tSNE results, clustering results and aggregated signal for CRE cluster:

```
# tSNE results
tsne <- clusterres[[1]]
head(tsne)

##                      [,1]       [,2]
## GSM1596840.bam 1.6190782  0.4044729
## GSM1596874.bam 2.3337455 -2.9356294
## GSM1596881.bam 3.5145605  0.5138395
## GSM1596942.bam 3.2520900 -1.2296946
## GSM1596944.bam 0.9427077 -0.7766757
## GSM1596961.bam 1.7692160 -0.5845746

# clustering results
cluster <- clusterres[[2]]
cluster

## GSM1596840.bam GSM1596874.bam GSM1596881.bam GSM1596942.bam GSM1596944.bam
##              1              1              1              1              1
## GSM1596961.bam GSM1597041.bam GSM1597096.bam SRR1779746.bam SRR1779805.bam
##              1              1              1              2              2
## SRR1779856.bam SRR1779874.bam SRR1779956.bam SRR1779959.bam SRR1779973.bam
##              1              2              2              1              2
```

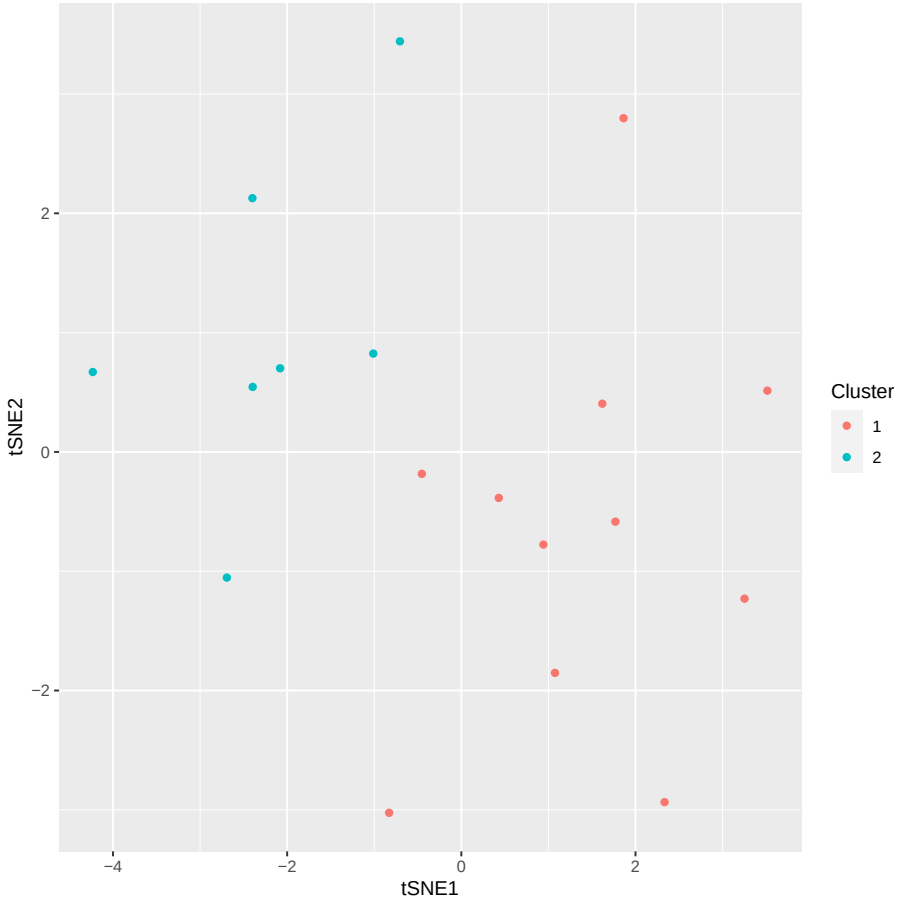
```
## SRR1780018.bam SRR1780020.bam SRR1780054.bam
##              1              2              2

# cell 'GSM1596840.bam' belongs to cluster 1, and cell 'SRR1779746.bam' belongs to cluster 2
# aggregated signal for CRE cluster
aggsig <- clusterres[[3]]
aggsig[1:3,1:3]

##    GSM1596840.bam GSM1596874.bam GSM1596881.bam
## 7       0.0000000      0.0000000      0.3607243
## 9       0.0000000      0.0000000      0.0000000
## 10      0.6758744      0.8080204      0.4568552
```

We can use the following code to draw the tSNE plot and mark the clusters.

```
library(ggplot2)
plotdata <- data.frame(tSNE1=tsne[,1],tSNE2=tsne[,2],Cluster=as.factor(cluster))
ggplot(plotdata,aes(x=tSNE1,y=tSNE2,col=Cluster)) + geom_point()
```



5 Run SCATE

The function `SCATE` will reconstruct signals for individual CREs. For example, below code performs SCATE using the scATAC-seq data and clustering results from the previous clustering step. For demonstration purpose, here number of CRE clusters is set to be 5000 (clunum=5000) to allow faster speed. In real application, clunum is recommended to set to NULL so SCATE will automatically choose the optimal number of clusters.

```
res <- SCATE(satac,genome="hg19",cluster=cluster,clusterid=NULL,clunum=5000,ncores=10,verbos

## [1] "Preparing data"
## [1] "Fitting model"
## [1] "Preparing data"
## [1] "Fitting model"
## [1] "Generating results"
```

```
# check the 10000-10005th row of the matrix
res[10000:10005,]
```

```
##                                1        2
## chr1_2009200_2009399 1.6157410 1.676166
## chr1_2009400_2009599 1.4959366 1.541986
## chr1_2009600_2009799 1.8051788 1.886171
## chr1_2009800_2009999 1.9564854 2.018323
## chr1_2010000_2010199 1.7938793 1.819427
## chr1_2010200_2010399 0.7672063 0.805896
```

If users only want to perform SCATE in a subset of clusters, they can set `clusterid` to be a subset of clusters. For example setting `clusterid=c(1,2)` will let SCATE to only run in cluster 1 and 2. `ncores` sets the number of computing cores to run SCATE in parallel. A larger number will result in less computational time, but will use more computational resources. If SCATE is run on a personal computer, ncores should be set to a small number such as 3. If `ncores=NULL`, all available cores will be used for computing. Note that currently ncores are forced to be 1 on Windows computers. If verbose is TRUE, the current progress will be displayed. If verbose is FALSE the information will be suppressed.
The output is matrix of reconstructed signals. The number of rows of the matrix is the same as the number of bins in the genome. The number of columns is the same as the number of clusters in which SCATE is performed. The column names indicate the cluster id.
Users can also supply their own cell cluster. An example can be seen below. `usercellcluster` specifies the cluster by users. Note that `usercellcluster` needs to be named to correspond with the list of GRanges.

```
# use similar ways to construct the cluster
usercellcluster <- rep(1:2,each=9)
names(usercellcluster) <- names(satac)
# check the contents of the cluster
usercellcluster
```

```
## GSM1596840.bam GSM1596874.bam GSM1596881.bam GSM1596942.bam GSM1596944.bam
##              1              1              1              1              1
## GSM1596961.bam GSM1597041.bam GSM1597096.bam SRR1779746.bam SRR1779805.bam
##              1              1              1              1              2
## SRR1779856.bam SRR1779874.bam SRR1779956.bam SRR1779959.bam SRR1779973.bam
##              2              2              2              2              2
## SRR1780018.bam SRR1780020.bam SRR1780054.bam
##              2              2              2
```

```
userclusterres <- SCATE(satac,genome="hg19",cluster=usercellcluster,clunum=5000,ncores=10,ve
```

Function `extractfeature` can be used to extract signals for a subset of genomic region of interest. For example if we want to extract signal for all bins that are within chromosome 5 50000-50300 and 50700-51000, we can first prepare the following region data:

```
region <- data.frame(chr=c('chr5','chr5'),start=c(50000,50700),end=c(50300,51000))
region
```

```
##    chr start   end
## 1 chr5 50000 50300
## 2 chr5 50700 51000
```

Region is a data frame with three columns. First column is chromosome name, second column is starting position, and third column is the ending position. Function `extractfeature` is then called in this way. `mode` can be either 'overlap' to include all bins that overlap with the given `region`, or be 'nearest' to include the nearest bin for each region.

```
extractres <- extractfeature(res,region,mode='overlap')
extractres
```

```
##                          1        2
## chr5_50000_50199 1.211534 1.220932
## chr5_50200_50399 1.363037 1.439014
## chr5_50600_50799 1.399362 1.492725
## chr5_50800_50999 1.425771 1.623218
## chr5_51000_51199 1.270912 1.483376
```

`extractfeature` also allows the extracted features to be saved to BED files that can be uploaded to genome browser (e.g. UCSC genome browser):

```
extractres <- extractfeature(res,region,mode='overlap',folder='destination folder')
```

In this example, a set of BED files will be saved to destination folder. Each BED file corresponds to a cluster and has the signal and genomic location for each bin. The name of the BED file corresponds to the name of the cluster.

6 Peak Calling

The function `peakcall` will perform peak calling on the SCATE results. For example, we can call peaks for the first cluster:

```
peakres <- peakcall(res)
# check the result for the first cluster
head(peakres[[1]])

##        chr    start      end FDR   Signal
## 8906 chr20 26188600 26190799   0 7.823961
## 5854 chr11 62607800 62609999   0 7.643640
## 6411 chr12 53772800 53774799   0 7.288096
## 9623 chr19 42772000 42773799   0 7.245268
## 9559 chr19 39339600 39341799   0 7.241847
## 9852 chr19 55850000 55851599   0 7.237216
```

For each cluster, the output is a data frame with five column. The first three columns tell the location of the peak. They are chromosome name, starting location and ending location respectively. The fourth column is the FDR of the peak, and the fifth column is the signal of the peak. Peaks are already ordered by FDR and then by signal.
Use the following code to save the peak results for the first cluster as BED file.

```
write.table(peakres[[1]],file='your file.bed',sep='\t',quote=F,col.names = F,row.names = F)
```

7 Wrapper function

The whole process of reading in bam, clustering cell, and performing SCATE is included in the `SCATEpipeline` function using following wrapper function of SCATE pipeline:

```
piperes <- SCATEpipeline(bamlist,genome="hg19",cellclunum=2,CREclunum=5000,perplexity=5,ncor
# get the cell cluster results, same as calling 'cellcluster' function.
cluster <- piperes[['cellcluster']]
# get the SCATE outputs, same as calling 'SCATE' function.
SCATEres <- piperes[['SCATE']]
# get the peak calling results, same as calling 'peakcall' function.
peakres <- piperes[['peak']]
```

As before, `bamlist` stores the locations to bam files and `genome` is 'hg19' or 'mm10'. The output is a list with three components. The first component is the cell clustering results, same as the one returned by `cellcluster`. The second component is the SCATE results, same as the one returned by `SCATE`. The third component is the peak calling results, same as the one returned by `peakcall`.
One can extract the region using the `extractfeature`

```
extractres <- extractfeature(piperes[['SCATE']],region,mode='overlap',folder='destination fo
```

8 Add new bulk samples or CRE to human and mouse databases

Users can add new bulk samples or CRE to human and mouse databases. This can be easily done with the following command:

```
makedatabase(datapath,savepath,bamfile=bamfile,cre=cre,genome='hg19')
```

Here, `datapath` is the path to the data package folder (e.g. myfolder/hg19/). User must first download the data package to use this function. The data package for hg19 and mm10 can be downloaded from http://jilab.biostat.jhsph.edu/projects/scate/hg19.zip or http://jilab.biostat.jhsph.edu/projects/scate/mm10.zip. The compressed file should be unzipped.
`savepath` is the path to save the generated database. e.g. myfolder/database.rds.
`bamfile` is the location of bulk DNase-seq bamfiles. The format is the same as `bamlist`
`cre` is a dataframe of new CRE sites to be added to the database. First column: chromosome name. Second column: start position. Third column: end position. The format is the same as `region` argument in function `extractfeature`.
`genome` should be either 'hg19' or 'mm10'. Default is 'hg19'.
The function will generate a database file in 'savepath', which can then be passed to SCATE main function. See the manual page of SCATE function for how to include user's own database. The function will take around 1-2 days to finish.

9 Build database from scratch

Users can also build database from scratch using bulk samples or pooled single-cell ATAC-seq samples (pseudobulks). This can be easily done with the following command:

```
makedatabase(datapath=NULL,savepath,bamfile=bamfile,cre=cre,genomerange=genomerange)
```

`savepath` is the path to save the generated database. e.g. myfolder/database.rds.
`bamfile` is the location of bulk DNase-seq bamfiles or pooled single-cell ATAC-seq bam files. The format is the same as `bamlist`
`cre` is a dataframe of new CRE sites to be added to the database. First column: chromosome name. Second column: start position. Third column: end position. The format is the same as `region` argument in function `extractfeature`.
`genomerange` is a data frame with two columns. First column is the chromosome and second column is the length of the genome. Example is https://genome.ucsc.edu/goldenpath/help/hg19.chr

The function will generate a database file in 'savepath', which can then be passed to SCATE main function. See the manual page of SCATE function for how to include user's own database. The function will take around 1-2 days to finish.

10 Use database build by users in SCATE

Once the databases are built, users can run the whole pipeline with the new database:

```
piperes <- SCATEpipeline(bamlist,datapath='path to new database')
```

The following code gives example of running each individual step with the new database. Note that the new database only needs to be specified in `cellcluster` and `SCATE`.

```
clusterres <- cellcluster(satac,datapath='path to new database',clunum=2,perplexity=5)
cluster <- clusterres[[2]]
res <- SCATE(satac,datapath='path to new database',cluster=cluster,clusterid=NULL)
```

11 Session Info

```
sessionInfo()
```