FLCNA: A statistical learning method for simultaneous copy number estimation and subclone clustering with single cell sequencing data

Fei Qin, Guoshuai Cai, Feifei Xiao

Last updated: 04/20/2022

1. Introduction to the FLCNA method

We developed the FLCNA method based on a fused lasso model to detect copy number aberrations (CNAs) and identify subclones simultaneously. To capture the biological heterogeneity between potential subclones, we developed the FLCNA method which is capable of subcloning, and simultaneously detecting breakpoints with scDNA-seq data. First, procedures including quality control (QC), normalization, logarithm transformation are used for pre-processing of the datasets. Subclone clustering is achieved based on a Gaussian Mixture Model (GMM), and breakpoints detection is conducted by adding a fused lasso penalty term to the typical GMM model. Finally, shared CNA segments in each cluster are clustered into three different CNA states (deletion, normal/diploid and duplication) using a GMM-based clustering strategy. The framework of the FLCNA method is summarized and illustrated in Figure 1.

2. Installation

```
library(devtools)
install_github("FeifeiXiaoUSC/FLCNA")
```

3. Bioinformatic pre-processing

For public data from NCBI SRA, starting with SRA files, FASTQ files can be generated with Fastq-dump from SRA-Toolkit, and then aligned to NCBI hg19 reference genome and converted to BAM files. For the 10× Genomics datasets, we need to demultiplex the original integrated BAM file into separate BAM files. Raw read depth of coverage data are generated from the BAM files with bin size 100kb. SCOPE R package can be utilized for generating coverage data, mappability and GC content. Specifically, get_bam_bed() can be used for generating bed files. get_coverage_scDNA() function can be applied for computing the depth of coverage for each cell and each marker. get_mapp() and get_gc() function can be used to calculate mappability and GC content, respectively.

4. Quality Control

FLCNA_QC() R function can be ued to remove samples/cells with low proportion of reads and bins that have extreme GC content (less than 20% and greater than 80%) and low mappability (less than 0.9) to

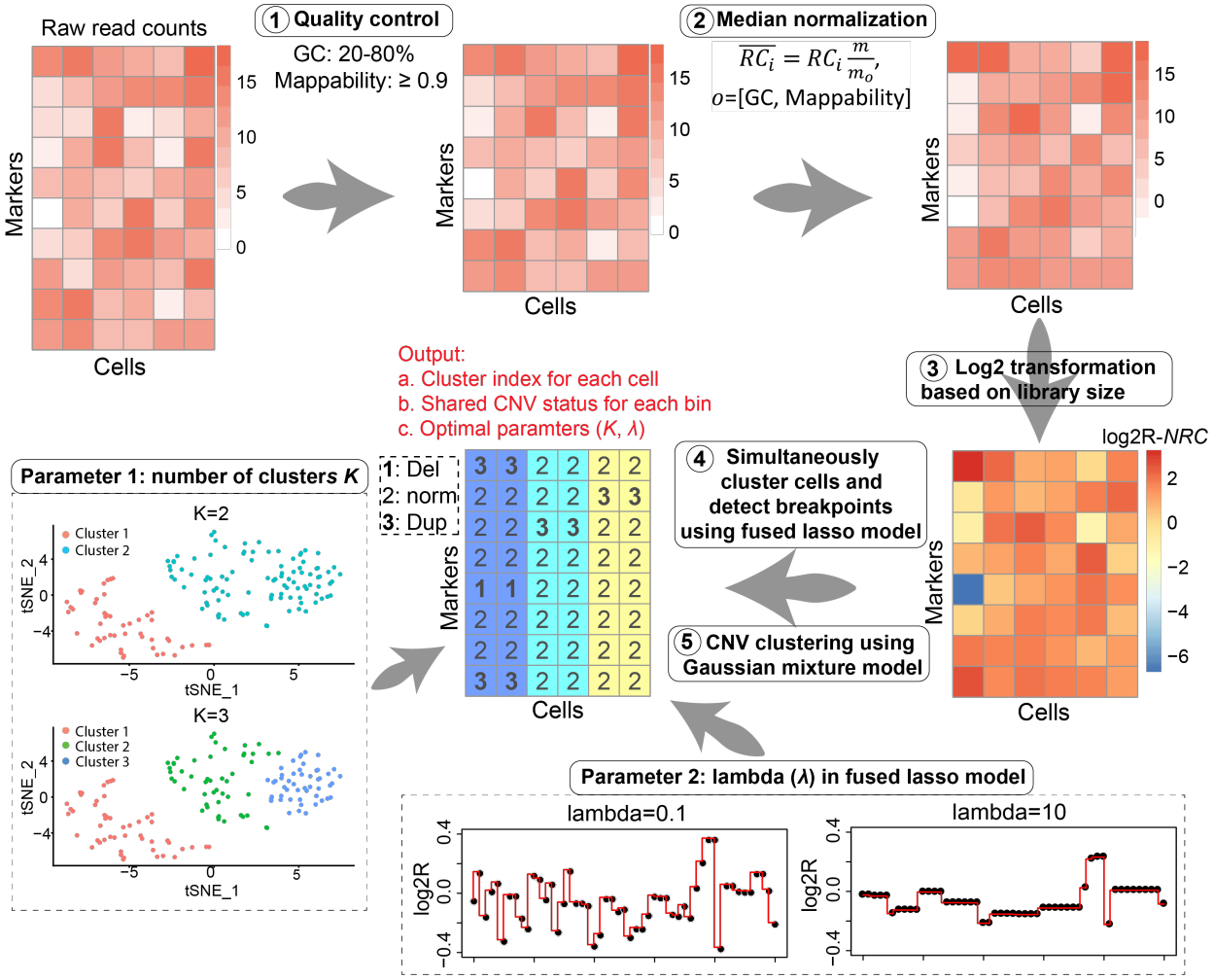

Figure 1: FLCNA framework

reduce artifacts.

```
# The example data have 2,000 markers and 200 cells.
library(FLCNA)
data(Example_data_2000)
data(Example_ref_2000)
RD <- Example_data_2000
dim(RD)
```

```
## [1]  200 2000
```

```
ref <- Example_ref_2000
head(ref)
```

```
## GRanges object with 6 ranges and 2 metadata columns:
##       seqnames          ranges strand |        gc      mapp
##          <Rle>       <IRanges>  <Rle> | <numeric> <numeric>
##   [1]     chr1 2000001-2100000      * |     56.96  0.984862
##   [2]     chr1 2800001-2900000      * |     57.94  0.992544
##   [3]     chr1 2900001-3000000      * |     55.43  0.984850
##   [4]     chr1 3000001-3100000      * |     56.60  0.995182
##   [5]     chr1 3100001-3200000      * |     58.16  0.989534
##   [6]     chr1 3200001-3300000      * |     56.83  0.973831
##   -------
##   seqinfo: 24 sequences from hg38 genome
```

```
QCobject <- FLCNA_QC(Y_raw=t(RD), ref_raw=ref,
                     mapp_thresh = 0.9,
                     gc_thresh = c(20, 80))
```

```
## Removed 0 samples due to failed library preparation.
```

```
## Removed 95 samples due to failure to meet min coverage requirement.
```

```
## Excluded 0 bins due to extreme GC content.
```

```
## Excluded 0 bins due to low mappability.
```

```
## There are 105 samples and 2000 bins after QC step.
```

5. Normalization

A two-step median normalization approach is implemented to remove the effect of biases from the GC-content and mappability. We further calculate the ratio of normalized RC and its sample specific mean, and the logarithm transformation of this ratio (log2R-NRC) is used in the main step of the FLCNA method. FLCNA_normalization() R function is used for the normalization.

```
log2Rdata <- FLCNA_normalization(Y=QCobject$Y, gc=QCobject$ref$gc, map=QCobject$ref$mapp)
```

6. Simultaneous CNA detection and subclone clustering

Subclone clustering is achieved based on a GMM, and breakpoints detection is conducted by adding a fused lasso penalty term to the typical GMM model. FLCNA() R function can be used for the CNA detection and simultaneous subclone clustering. There are two hyperparameters to be pre-defined in the FLCNA method, including the number of clusters K and the tuning parameter lambda. The tuning hyperparameter lambda is used to control the overall number of change points that less change points tend to be generated with larger lambda value. To find the optimal values of K and lambda, we use a BIC-type criterion, and the model with smallest BIC value is selected as the optimal model.

```
# K: The number of clusters.
# lambda: The tuning parameter in the penalty term, the default is 3.
output_FLCNA <- FLCNA(K=c(4,5,6), lambda=3, Y=data.matrix(log2Rdata))
```

```
# The number of clusters in the optimal model
output_FLCNA$K.best
```

```
## [1] 5
```

```
# The estimated mean matrix for K clusters
output_FLCNA$mu.hat.best[,1:11]
```

```
##              [,1]         [,2]         [,3]         [,4]         [,5]
## [1,]  0.007205948  0.007207692  0.007207827  0.007207865  0.007209344
## [2,] -0.003407598 -0.003407066 -0.003414070 -0.003414967 -0.003415942
## [3,]  0.007120851  0.007121311  0.007128402  0.007128396 -0.003978781
## [4,]  0.007187759  0.007188329  0.007195617  0.007199059  0.006410679
## [5,]  0.002242498  0.002247932  0.002249631 -0.121560933 -0.121560775
##              [,6]         [,7]         [,8]         [,9]      [,10]      [,11]
## [1,]  0.007280049  0.007284358  0.007287120  0.007290247 0.02953363 0.02953813
## [2,] -0.003415992 -0.032819296 -0.032819567 -0.032817131 0.04861164 0.04861432
## [3,] -0.003976100 -0.003976030 -0.003975317  0.014545825 0.01454448 0.01453701
## [4,] -0.004333336 -0.004332628 -0.004331771 -0.004323489 0.05664757 0.05664904
## [5,] -0.121558954 -0.121558281 -0.136911637 -0.136903277 0.05805697 0.05805765
```

```
# The cluster index for each cell
output_FLCNA$s.hat.best
```

```
##   [1] 3 2 2 1 4 5 2 1 2 3 1 2 4 1 3 2 4 3 1 2 4 4 5 3 1 1 3 3 4 1 3 5 5 4 2 2 1
##  [38] 1 3 5 3 5 5 5 5 4 1 1 5 2 3 5 1 2 3 4 3 2 1 2 4 5 4 1 5 4 1 5 1 2 4 5 1 4
##  [75] 5 2 5 1 5 5 3 5 1 4 1 4 4 1 1 2 3 4 2 5 1 1 2 4 5 4 1 4 5 1 2 5 4 4 4 4 3
## [112] 1 4 3 1 1 5 3 4 4 4 1 4 4 4 4 2 4 1 4 4 1 1 5 4 1 5 2 1 4 2 4 4 5 4 3 1 4
## [149] 2 5 4 5 1 1 5 1 4 1 5 3 2 5 1 1 5 3 2 1 5 1 5 2 1 2 2 5 2 3 2 3 2 1 2 4 3
## [186] 1 5 5 2 3 1 3 2 1 2 5 5 4 4 4
```

7. CNA clustering

After the mean vector is estimated for each cluster, we locate and quantify all the change points, and identify segments that share the same underlying copy number profile. CNA.out() R function is used for the clustering of candidate CNAs. Change-point can be identified from the estimate of mean vector where

for the marker before and after the change point show different values. Typically, different CNA states are required to be assigned for each segment to help locate significant CNA signatures. For each cluster, to assign the most likely copy number state for each segment, we further implemented a GMM-based clustering strategy for CNA clustering based on the estimate of mean vector. Each segment will be classified using a three-state classification scheme with deletion, normal/diploid and duplication.

```
# mean.matrix: The cluster mean matrix estimated from FLCNA R function.
# cutoff: Cutoff value to further control the number of CNAs, the larger value of cutoff,
# the smaller number of CNAs. The default is 0.35.
# L: Repeat times in the EM algorithm, defaults to 100.
CNA.output <- CNA.out(mean.matrix = output_FLCNA$mu.hat.best, ref=ref, cutoff=0.35, L=100)
CNA.output
```

```
## [[1]]
##   state     start       end  chr width_bins
## 1   del  34400001  37600001 chr1         24
## 2   del  72500001  77900001 chr1         34
## 3   del 170700001 173400001 chr1         27
## 4   del 225200001 230400001 chr1         26
## 5   del  32000001  36100001 chr2         31
##
## [[2]]
##   state     start       end  chr width_bins
## 1   del  39600001  47800001 chr1         30
## 2   del 102000001 106100001 chr1         22
## 3   del 190700001 193300001 chr1         26
## 4   del 241300001 246700001 chr1         26
## 5   del  49600001  53900001 chr2         34
##
## [[3]]
##   state     start       end  chr width_bins
## 1   del  28200001  31000001 chr1         24
## 2   del  55600001  59700001 chr1         32
## 3   del 117400001 151800001 chr1         30
## 4   del 203100001 205500001 chr1         24
## 5   del 237900001 243600001 chr1         36
## 6   del  44800001  50800001 chr2         36
##
## [[4]]
##   state     start       end  chr width_bins
## 1   del  39200001  46500001 chr1         25
## 2   del  93800001  98100001 chr1         32
## 3   del 187100001 190600001 chr1         35
##
## [[5]]
##   state     start       end  chr width_bins
## 1   del  46500001  49700001 chr1         25
## 2   del 112900001 115600001 chr1         27
## 3   del 182300001 187100001 chr1         30
## 4   del 214800001 218400001 chr1         35
## 5   del  26100001  28600001 chr2         25
```