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: 03/09/2023

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, based on these shared breakpoints in each cluster, candidate CNA segments for each cell were clustered into different CNA states 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 Sequence Read Archive (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

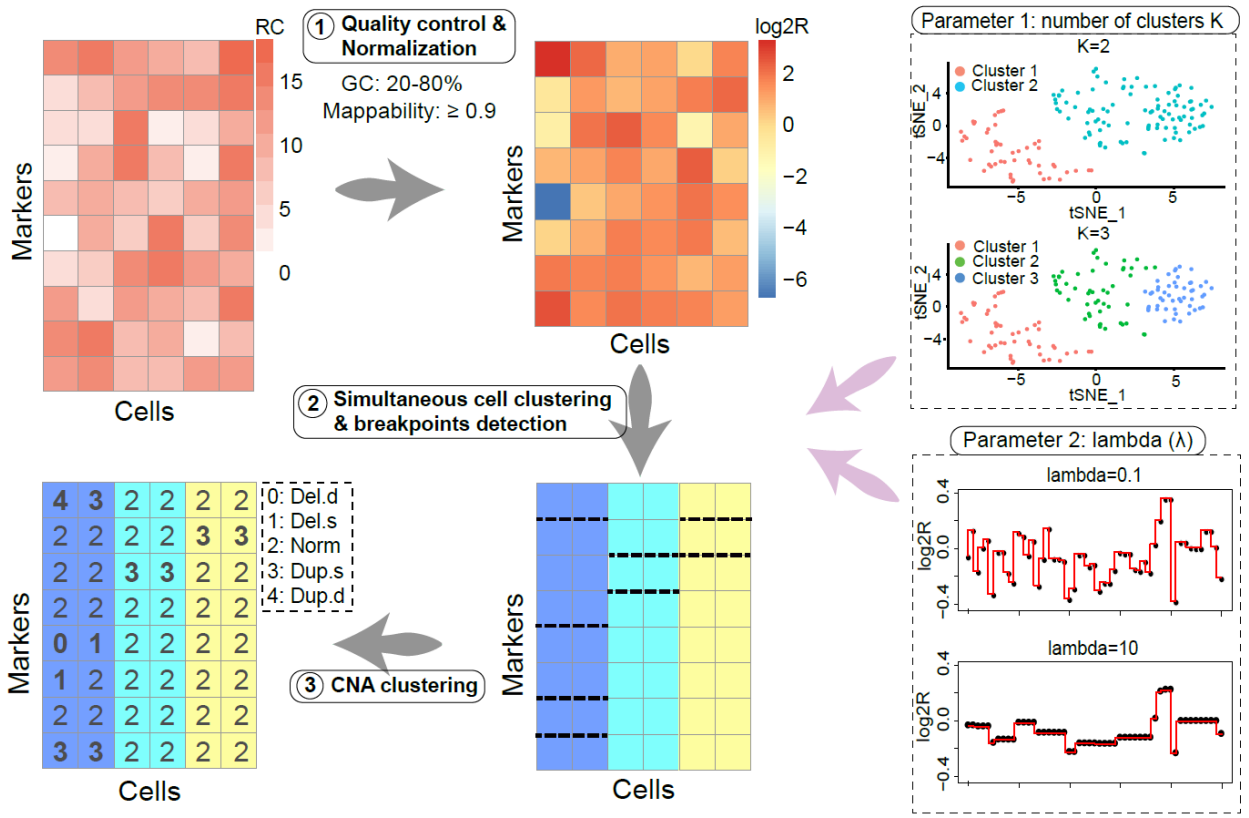

Figure 1: FLCNA framework

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. Data

To help explain how FLCNA can be used, we utilized a TNBC dataset from triple-negative breast cancer (TNBC) patients in NCBI SRA (SRP114962). TNBC displays extensive intra-tumor heterogeneity and frequently develops resistance to neoadjuvant chemotherapy (NAC) treatment. The KTN126 patient was used in this vignette, where tumor cells were only reported in the pre-treatment samples. Cells were sequenced at two time points (pre- and mid/post-treatment) with 93 cells (46 pre- and 47 post-treatment) in the KTN126 patient. Raw read depth of coverage data were generated from the BAM files with bin size 100k.

5. Quality Control

FLCNA_QC() R function can be used to remove bins that have extreme GC content (less than 20% and greater than 80%) and low mappability (less than 0.9) to reduce artifacts.

```
# The example data have 3,000 markers and 93 cells.
library(FLCNA)
data(KTN126_data_3000)
data(KTN126_ref_3000)
RD <- KTN126_data_3000
dim(RD)
```

```
## [1] 3000   93
```

```
ref <- KTN126_ref_3000
head(ref)
```

```
## GRanges object with 6 ranges and 2 metadata columns:
##       seqnames        ranges strand |        gc      mapp
##          <Rle>     <IRanges>  <Rle> | <numeric> <numeric>
##   [1]     chr1      1-100000      * |     38.21  0.275225
##   [2]     chr1 100001-200000      * |     33.76  0.119393
##   [3]     chr1 200001-300000      * |     15.78  0.141733
##   [4]     chr1 300001-400000      * |     33.08  0.115785
##   [5]     chr1 400001-500000      * |     32.38  0.155596
##   [6]     chr1 500001-600000      * |     35.18  0.206961
##   -------
##   seqinfo: 24 sequences from hg19 genome
```

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

```
## Excluded 233 bins due to extreme GC content.
```

```
## Excluded 210 bins due to low mappability.
```

```
## There are 93 samples and 2572 bins after QC step.
```

6. 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) 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)
```

7. 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(3,4,5), lambda=3, Y=log2Rdata)
```

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

```
## [1] 4
```

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

```
##            [,1]       [,2]       [,3]       [,4]       [,5]        [,6]
## [1,] -1.3428226 -0.1932012 -0.1931969 -0.3443138 -0.8830260 -0.11986545
## [2,] -1.2005804 -0.3594904 -0.3594859 -0.6050274 -0.6075824 -0.25178873
## [3,] -1.0563435 -0.2089481 -0.2089425 -0.4364831 -0.7848446 -0.04731857
## [4,] -0.5065039 -0.5040336 -0.3992638 -0.3992600 -0.3992577 -0.39925142
##             [,7]        [,8]        [,9]       [,10]       [,11]
## [1,] -0.09089903 -0.09089832  0.11729189  0.11729300  0.15844515
## [2,] -0.04883023 -0.04882879 -0.04882691 -0.04882715 -0.04882654
## [3,] -0.04731449 -0.04731633  0.19133971  0.19133986  0.19133914
## [4,] -0.39925090 -0.39925766 -0.04146510 -0.04146562 -0.04243215
```

```
# The cluster index for each cell
output_FLCNA$s.hat.best[1:20]
```

```
##  [1] 1 1 1 1 1 1 1 1 1 2 1 1 2 4 3 2 2 2 1
```

```
table(output_FLCNA$s.hat.best)
```

```
##
##  1  2  3  4
## 62 18 11  2
```

8. 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 for each cell. 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 in each cell to help locate significant CNA signatures. A GMM-based clustering strategy is implemented for CNA clustering using the normalized read counts data (i.e., log2R). Segments sharing similar intensity levels (i.e., the log2R values) in a cell are identified as the ones with same copy number states. Each segment is classified using a five-state classification scheme with deletion of double copies (Del.d), deletion of a single copy (Del.s), normal/diploid, duplication of a single copy (Dup.s) and duplication of double copies (Dup.d).

```
# mean.matrix: The cluster mean matrix estimated from FLCNA R function.
# cutoff: Cutoff value to further control the number of CNAs,
#         the larger 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$mu.hat.best, LRR=log2Rdata,
                      Clusters=output_FLCNA$s.hat.best, ref=ref,
                      cutoff=0.35, L=100)
head(CNA.output)
```

```
##   sampleID samplename Cluster  chr start end start.coor end.coor width_bins
## 1        1 SRR5964097       1 chr1     2   6     100001   600001          5
## 2        1 SRR5964097       1 chr1     6  14     500001  1400001          9
## 4        1 SRR5964097       1 chr1    15  27    1400001  2700001         13
## 5        1 SRR5964097       1 chr1    81 140    8000001 14000001         60
## 6        1 SRR5964097       1 chr1   212 214   21100001 21400001          3
## 7        1 SRR5964097       1 chr1   473 700   47200001 70000001        228
##   state
## 1 Del.d
## 2 Del.s
## 4 Del.s
## 5 Del.s
## 6 Del.d
## 7 Del.s
```