**Genetic Location Identification via Bionano**

**Version 1.6**

**User manual**

**Chunlong Chen, Weitao Wang**

**2020-09-30**

**Contents**

Prerequisites

1. JDK version 1.8
2. R is required for peak calling
3. About 7GB of available RAM
4. Steps in this document assumes that your current working directory is ‘example’ and that the ‘replicon’ executable is in the parent directory.

Introduction

This is an extension software analyzing data based on Bionano high throughput single‑molecule imaging platform determination of any biological process associated with fluorescently labeled DNA, for example: DNA replication, DNA recombination, DNA repair. It provides the analysis on each single molecule as well as the frequency occurred in population‑based data along the genome and the given genomic positions of interest.

Packages

Notice: Please pay attention that the parent directories should be the folder containing all corresponding files sharing the same filename (indicated by the “SampleName” parameter) besides suffix, i.e. the parent directories containing all the .bnx, .xmap, .rcmap, .qcmap or .txt input files and output files must be with the same file name beside suffix, and must be end with corresponding suffix for data format recognition.

1. AllRawDataRefining.jar

The input data are 4 kinds of basial format data in Bionano platform they are [bnx](https://bionanogenomics.com/wp-content/uploads/2018/04/30038-BNX-File-Format-Specification-Sheet.pdf) (raw labeling red signals data) , [xmap](https://bionanogenomics.com/wp-content/uploads/2017/03/30040-XMAP-File-Format-Specification-Sheet.pdf) (mapping data), qcmap (base calibration data) ,rcmap (reference data). The jar packages should be run under terminal. It will refine all information from these 4 input files, filter fibers and calculate the precise genomic positions of labelling signals on each filtered fiber. Normally, there are 2 channels, one stores the green signals for mapping fibers to reference, another is used to store the labelling red signals. Different experiments may have different choices for different channels to label. So the user need to set which channel used for labelling and which channel used for mapping.

Option:

-S: Could be 1 or 2 used for setting which channel is signal channel for labelling

-M: Could be 1 or 2 used for setting which channel is for mapping (should be different with -S)

-B: BNX parent directory

-X: XMP parent directory

-R: RCmap parent directory

-Q: QCmap parent directory

-O: Output directory

-SampleName: All SampleName seperated by "," delimiter, same as all filename in various parent path and without any suffix in this option

-WNS: Want no signal fibers or not. It could be “Y” or “N”, the default value is “Y”, if the user set this item as “N”, it means the output files won’t contain no-signal-fibers.

-WGI: Want green information for fiber with red signals. It could be “Y” or “N”, the default value is “N”, if the user set this item as “Y”, it means it will list out all green mapping signal’s intensity, SNR and positions for the fibers with red signals.

-WROG: Want red labelling signals outsides the range from first to last green mapping signal. The default is “N”, if the user set this item as “Y”, it means there will be an extra output files record all the fibers with Red\_Out\_Small\_Count / Red\_Out\_Big\_Count (red signal number whose position smaller / bigger than the first / last mapping green signal). The format is like below.  
Chromosome FiberStart FiberEnd FiberID Red\_Out\_Small\_Count Red\_Out\_Big\_Count total\_red\_number total\_green\_number

**An Example (Quick start):**

java -jar AllRawDataRefining\_1G2R.jar

-B /Volumes/WWT/Final-Version/1905/BNX/

-X /Volumes/WWT/Final-Version/1905/XMP/

-R /Volumes/WWT/Final-Version/1905/RCmap/

-Q /Volumes/WWT/Final-Version/1905/QCmap/

-O /Volumes/WWT/Final-Version/FilterHotDot/BaseCorrect/TXT/WithRed/WithHot/

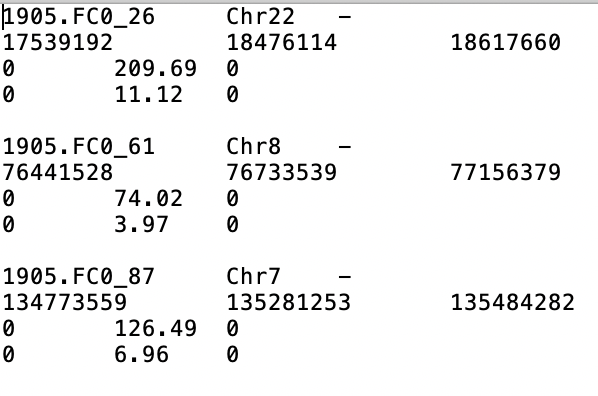
-SampleName 1905.FC1,1905.FC2,1905.FC0

-S 2

-M 1

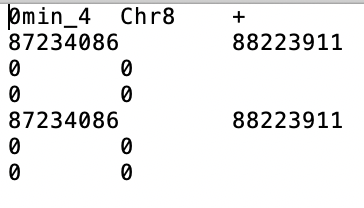
**The output files**

A .txt file records all refined calculation result about mapped fibers with signals of interest (in our case, ORM signals in red channel) after base calibration from raw data (qcmap,rcmap, xmp and bnx files). The information example is like below. Most of following analysis need it as input file.

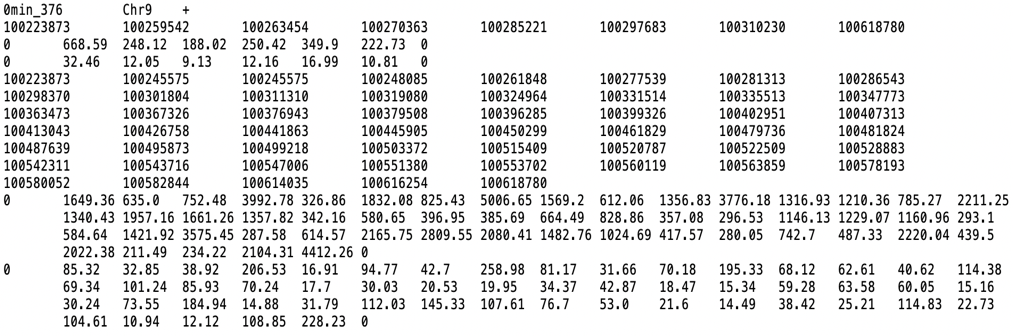
****

Each fiber record was organized by 4 lines. The 1st line’s first item is a record ID organized by Sample name and molecular ID from .bnx file separated by “\_”. The second line is the genomic position of all labeling signals and coordinate of 2 ends of fiber (The first position and last position). The 3rd and 4th line are corresponding SNR(signal noise ratio) and signal intensity from .bnx file too.

If -WGI item set as “Y” the output files will be like below.



Each fiber record was organized by 7 lines. The first 4 lines is same as .txt output file without “-WGI” parameter choice. And the 5th to 7th lines are the positions for green mapping signals, SNR and intensity. However, the original intention for this function is to compare the red signals and green signals intensity and distribution difference. So for all fibers without red signals just record the two ends of fibers same as red labelling signal lines. But for fiber with red signals the output will be like below.



The above plot has too much information to show, so it crowded into 19 lines. But in fact, there are only 7 lines. The lines for positions always start and end as two fiber’s ends. And correspondingly, the first and last values for lines record SNR and intensity always 0, no matter they are for red signals or green signals.

1. GenerateGTF\_ByAllDataRefining\_V2.jar

The input data needs .txt file(s) got by AllRawDataRefining.jar. It is used to generate a gtf-like file for each input .txt file, in order to be able to visualize the fibers together with their labelling signals on the IGV (or any other genomic browsers that can load gtf file).

Parameters:

-ID\_List: Sometimes we just want to generate the specific fibers to check their signals distribution, so here it is a direct path for a file record all interesting fibers’ ID

-I: The parent path of the input txt file(s)

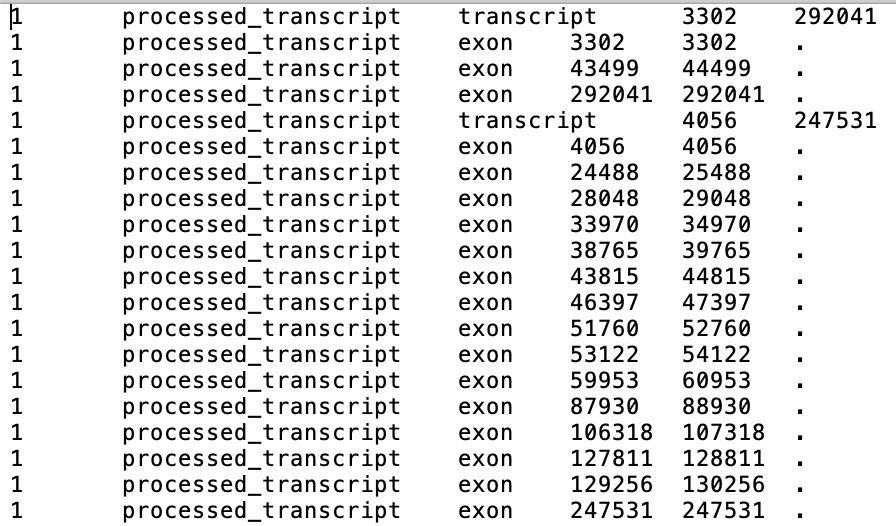
-O: The path of output gtf file(s)

-SampleName: the name of all txt file, multi files will be separated by “,” and the name of txt file can’t contain character “\_”, since it has been used as delimiter in script.

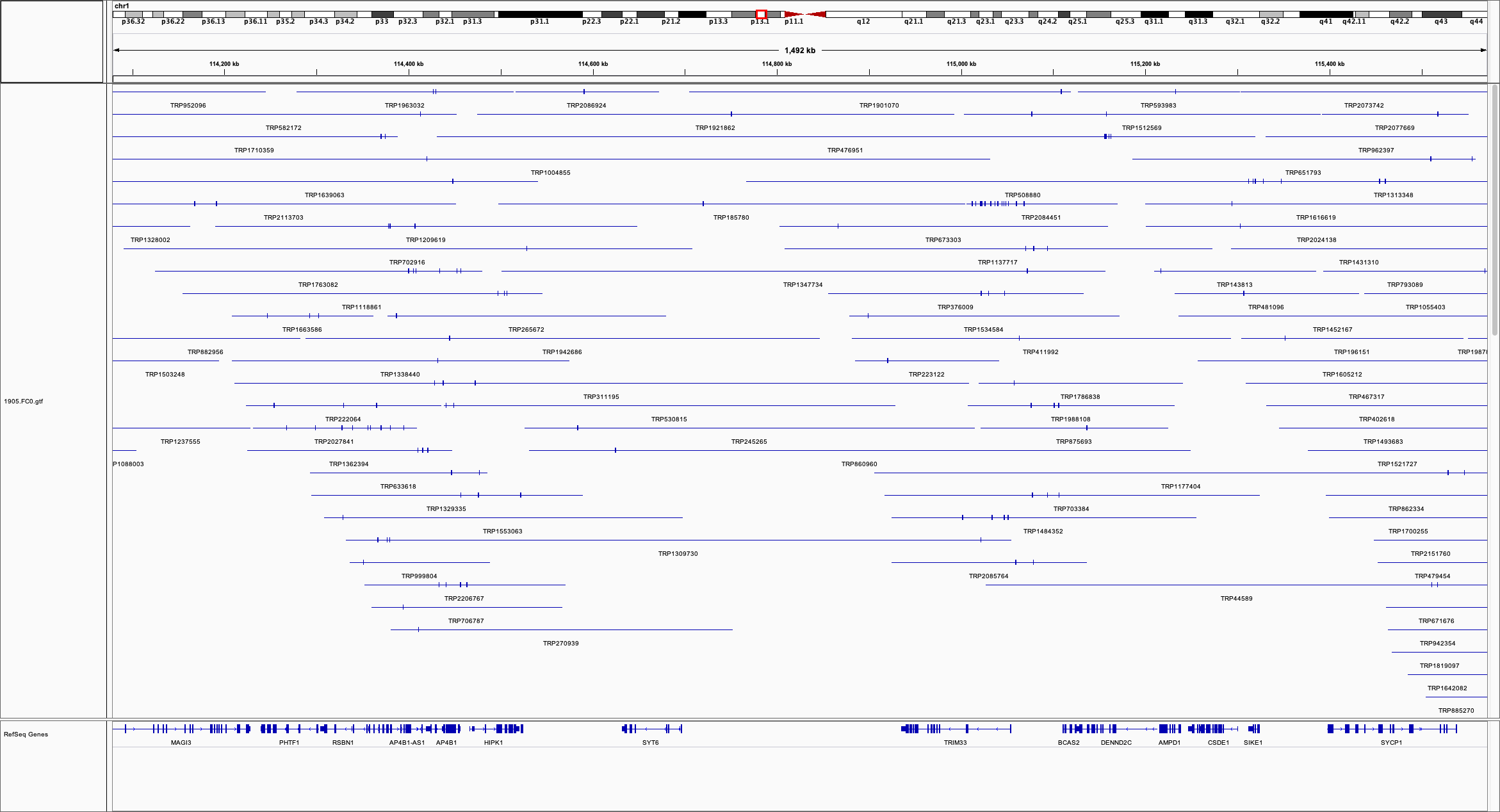
-Contain: The option could only be T when it shows up, and the default value is “false” when it is missing. It is used to filter out the fibers without any signal of interest, default is unfiltering.

**The output files**

A gtf-like file (.gtf) for each input .txt file.

****

Here, each fiber is labeled as a transcript and for data visualization in IGV. We labeled all labelling signals and 2 ends of each fiber as exons. The start and end coordinates for the start and end positions, respectively, of each fiber is identical (i.e. exon length equal to 1 nt). For the visualization reason, the start and end coordinates of each signal is set as its position – and +, respectively, 500 nt (i.e. exon length equal to 1 kb). As illustrated in the following figure, each line is a fiber and each little bar on the line corresponds to a labelling signal.



Fiber data in GTF format visualization under IGV

**An Example (Quick start):**

java -jar GenerateGTF\_ByAllDataRefining\_Reformat.jar

-I /Volumes/WWT/from\_dalila/Project/TXT/

-O /Volumes/WWT/from\_dalila/Project/IZ\_calling/GTF/Test/

-SampleName HCTRAD21CTR.txt,HCTRAD21KD.txt

-Contain T

-ID\_List /Volumes/WWT/from\_dalila/Project/Segment/Step4\_Segment/CTR\_ID.txt

Notice: The optional parameter is shown in red. The content for ID\_list option in CTR\_ID.txt is like below

Text

Description automatically generated

1. GetFiberCoordinate\_ByTXT.jar

The input data needs .txt file(s) got by AllRawDataRefining\_1R2G.jar or AllRawDataRefining\_1G2R.jar. It is used to generate fiber’s bed files.

Parameters:

-I: The parent path of the input txt file(s)

-O: The path of output file(s)

-SampleName: the name of all txt file, multi files will be separated by “,” and the name of txt file can’t contain character “\_”, since it has been used as delimiter in script.

**An Example (Quick start):**

java -jar GetFiberCoordinate\_ByTXT.jar

-I /Volumes/WWT/Final-Version/1807\_1802\_1905\_1708\_Merge/TXT/AlldataRefining/0min/

-O /Users/wwang/Desktop/ORM/GetFiberCoordinate\_ByTXT/Test

-SampleName 1905.FC0.txt,1905.FC1.txt

1. GetRedflagNumberInSlidingWindow\_ByAllDataRefining.jar

The input data needs .txt file(s) got by AllRawDataRefining\_1R2G.jar or AllRawDataRefining\_1G2R.jar. It is used to calculate the signal counts within sliding windows.

Parameters:

-I: The parent path of the input txt file(s)

-O: The path of output file(s)

-SampleName: the name of all txt file, multi files will be separated by “,” and the name of txt file can’t contain character “\_”, since it has been used as delimiter in script.

-REF: The input REF file for the reference chromosome length, unit nt, see below for an example (can find for example at UCSC genome browser):

chr1 249250621

chr2 243199373

chr3 198022430

…

Please pay attention, here the chromosome names must be the same as the chromosome names show up in .txt files (case sensitive, i.e. Chr and chr is different). If using chr23 and chr24 to represent chrX and chrY, respectively, in .txt file, the corresponding chromosome names in this REF file should be chr 23 and chr 24 instead of chrX and chrY.

-WS: window size, Unit kb

-SS: SlidingStep size, Unit kb

**An Example (Quick start):**

java -jar GetRedflagNumberInSlidingWindow\_ByAllDataRefining.jar

-REF /Users/wwang/Desktop/enrichment/ChrLength\_hg19.txt

-WS 10

-SS 1

-I /Volumes/WWT/Final-Version/1905/TXT/AllDataRefining/

-SampleName 1905.FC1.txt,1905.FC2.txt

-O /Users/wwang/Desktop/ORM/GetRedflagNumberInSlidingWindow\_ByAllDataRefining/Test

\*calculation of signal numbers within 10kb sliding window with 1kb step.

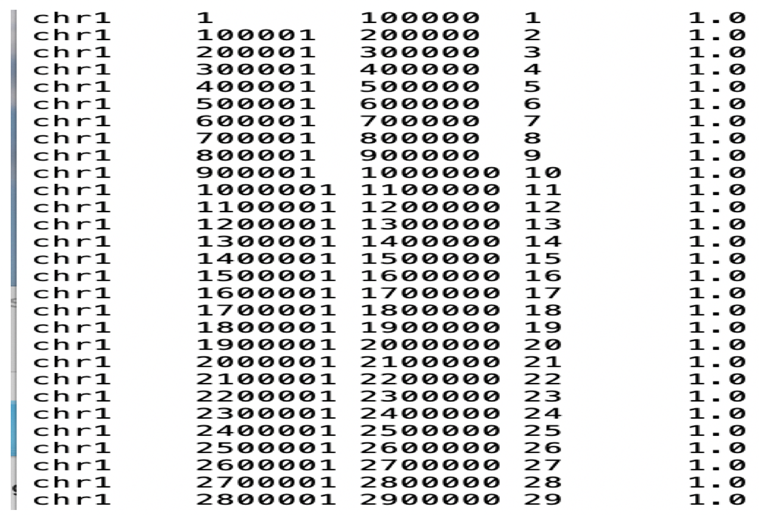
**Output format**

**The 8 columns in the output file corresponding to:**   
Chr Bin\_start Bin\_end Bin\_ID RedsignalNumber FiberNumber NormalizedSignalNumber FiberwithoutSignal  
  
where NormalizedSignalNumber = RedsignalNumber / FiberNumber , which is the signal count after fiber depth normalization.

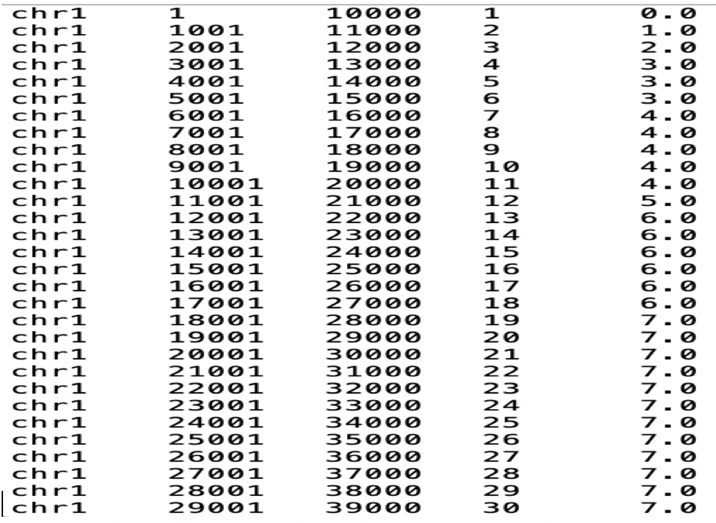
1. GetRedflagNumberInAdjacentWindow\_ByAllDataRefining.jar

It is similar as GetRedflagNumberInSlidingWindow\_ByAllDataRefining.jar but calculates signal number count within continuing non-overlapping windows.

**Output format**   
Non-overlapping adjacent Bins (in case window\_size=100kb)



Sliding Bins (in case window\_size=100kb, SlidingStep=1kb)



**An Example (Quick start):**

java -jar GetRedflagNumberInSlidingWindow\_ByAllDataRefining.jar

-REF /Users/wwang/Desktop/enrichment/ChrLength\_hg19.txt

-WS 10

-I /Volumes/WWT/Final-Version/1905/TXT/AllDataRefining/

-SampleName 1905.FC1.txt,1905.FC2.txt

-O /Users/wwang/Desktop/ORM/GetRedflagNumberInSlidingWindow\_ByAllDataRefining/Test

1. GetNewSegmentation\_AddSoloSignal.jar  
   This jar package is used for clustering neighbored signals based on the distance cutoff values set by user to get the final segments (i.e. ORM tracks in our case). It’s a two-step process: it clusters at first the close signals to get the primary segments then further fusion the close primary segments into the final segments. In order to get the proper cutoff values used in each step for different experiment data, this jar package also calculates the statistics on the distribution of distance between raw signals and the distance between primary segments once the first cutoff for adjacent signals being set. In addition, the users can choose whether including the single signal into the segmentation results (in case they have strong confidence on their data, which are real signals rather than technical noise) by setting -AddSolo options as T. If so, the single signal will become a solo segment with same start and end coordinate in the output file.  
     
   Notice: All output is not in order. If you wanted sorted bed files you can use the linux commend below to sort the output files

sort -n -k 1.4 -k 2 -k 3 input.bed > output.bed

Parameters:

-I: The parent path of the input txt file(s)

-O: The path of output file(s)

-SampleName: the name of all txt file, multi files will be separated by “,” and the name of txt file can’t contain character “\_”, since it has been used as delimiter in script.

-GetSignalDistance: if it is “T” the script will generate a document recording the distances between adjacent signals, and the default value is “false”.

-FC: The first cutoff value(s) to cluster the close signals. Multi cutoff values for different files could be separate by “,”. All signals with distance smaller than the corresponding cutoff will be clustered into a primary segment.

-GetSegmentDistance: if it is “T” the script will generate a document recording the distances between adjacent primary segments, and the default value is “false”. And you should put the -FC options prior to -GetSegmentDistance, since the primary segments needed for the analysis depend on the cutoff value set by -FC.

-SC: The second cutoff value(s) to cluster the close segments. Multi cutoff values for different files could be separate by “,”. All primary segments with distance smaller than the corresponding cutoff will be clustered into a final segment, and the script will generate the final bed file(s).

"-AddSolo": If it is “T”, the final result will contain all single signals (each as a solo signal segment) that are not merged into segments in previous steps. The default is “false”;

**Examples (Quick start):**

java -jar GetNewSegmentation\_AddSoloSignal.jar

-GetSignalDistance T

-I /Volumes/WWT/Final-Version/1905/TXT/AllDataRefining/

-SampleName 1905.FC1.txt,1905.FC2.txt

-O /Users/wwang/Desktop/ORM/GetNewSegmentation\_AddSoloSignal/Test

java -jar GetNewSegmentation\_AddSoloSignal.jar

-FC 16384,16384

-GetSegmentDistance

-I /Volumes/WWT/Final-Version/1905/TXT/AllDataRefining/

-SampleName 1905.FC1.txt,1905.FC2.txt

-O /Users/wwang/Desktop/ORM/GetNewSegmentation\_AddSoloSignal/Test

java -jar GetNewSegmentation\_AddSoloSignal.jar

-FC 16384,16384

-SC 32768,32768

-I /Volumes/WWT/Final-Version/1905/TXT/AllDataRefining/

-SampleName 1905.FC1.txt,1905.FC2.txt

-O /Users/wwang/Desktop/ORM/GetNewSegmentation\_AddSoloSignal/Test

(-AddSolo T)

1. GMM.R

This is an R package used for drawing gaussian mixture model distributions. In order to better set the cutoff values to cluster close signals and primary segments, user can do the gaussian mixture model classification. Most of time we classify into 3 distribution (i.e. corresponding to signals from the same event, signals from adjacent event, and noise, respectively) and choose the tail of first distribution as the cutoff value. The data of distance between signals or distance between primary segments come from jar package GetNewSegmentation\_AddSoloSignal.

The user needs to set the Input\_ParentPath, Output\_ParentPath, Names of all samples (the same as indicated in other packages) and the X axis’ range (i.e. limit for the X axis) for the plot.

1. Add\_FDI\_ToSegment.jar

-FDI\_N: the segments with negative FDI values and between 0 and this cutoff are filtered out.

-FDI\_P: the segments with positive FDI values and between 0 and this cutoff are filtered out.

-Signal\_N: The cutoff value for filtering the segments with negative FDI containing signals smaller than the cutoff

-Signal\_P: The cutoff value for filtering the segments with positive FDI containing signals smaller than the cutoff

-I-T: The parent path of the input txt file(s)

-I-B: The parent path of the input bed file(s)

-SampleName: the name of all input file, multi files will be separated by “,” and the name of txt file can’t contain character “\_”, since it has been used as delimiter in script. Please pay attention the .txt and .bed files should show up in pair. The .bed file is generated by script GetNewSegmentation\_AddSoloSignal. And the name of the pair should be same except suffix, for example Sample1.txt and Sample1.bed

-O: The path of output file(s)

**An Example (Quick start):**

Java -jar Add\_FDI\_ToSegment.jar

-FDI\_N -0.6

-FDI\_P 0.8

-Signal\_N 3

-Signal\_P 3

-I-T /Volumes/WWT/Final-Version/1905/TXT/AllDataRefining/

-I-B /Volumes/WWT/Final-Version/1905/Bed/

-SampleName 1905.FC1,1905.FC2

-O /Users/wwang/Desktop/ORM/Add\_FDI\_ToSegment/Test/

**Functions specific for DNA replication data**

1. GetS50Timing.jar

Replication timing data can be added into the analysis. For example, in our study, we include the replication timing obtained by Repli-Seq, defined as S50, the fraction of the S phase (0<S50<1) at which 50% of DNA is replicated in a defined region, which is computed by a linear interpolation of enrichment values of Repli-Seq signals obtained within six S phase compartments. It can reflect the temporal order of replication along the genome. To run this package, you need to provide S50 timing along the genome. It is 23 one column files which recorded all the S50 value in 1kb sliding step for 100kb window along the genome.

Parameters:

-I: the parent path of the bed file(s) that you want to add S50 values. Please pay attention the input bed file(s) should be sort by chromosome order, and the chromosome names need to match between the bed file(s) and the S50 file.

-O: the parent path of the output file(s)

-S50: the parent path of the document containing the one column S50 values along the genome

-SampleName: the name of all input file, multi files will be separated by “,” and the name of bed file can’t contain character “\_”, since it has been used as delimiter in script.

**An Example (Quick start):**

Java -jar Add\_FDI\_ToSegment.jar   
 -I /Volumes/WWT/Final-Version/1905/Bed/sort   
 -O /Users/wwang/Desktop/ORM/GetS50Timing/Test   
 -S50 /Users/wwang/Desktop/cluster/S50\_1kbSlide\_100kbWindow/   
 -SampleName 1905.FC0.bed,1905.FC1.bed

1. Add\_DeltaRFD\_ToSegment.jar

In DNA replication field, the RFD (Replication Fork Directionality) curve obtained by OK‑Seq method can be used to call initial zones. It is calculated by the numbers of Okazaki fragments corresponding to leftward (L) and rightward (R) replication forks, RFD=(R-L)/(R+L). The RFD values that we used are computed in 1 kb adjacent non-overlapping windows. When crossing replication origins, RFD curve shows a sharp increasing trend, so the difference of RFD values in the end and start position of a replication origin, called it Delta\_RFD values, are positive. The larger the Delta\_RFD values is the higher firing efficiency of corresponding replication origin. See ([REF](https://www.nature.com/articles/ncomms10208#Sec14)) for more detail.  
  
-I: the parent path of the bed file(s) that you want to add Delta\_RFD values. Please pay attention the input bed file(s) should be sort by chromosome order, and the chromosome names need to match between the bed file(s) and the RFD file.

-O: the parent path of the output file(s)

-RFD: the parent path of the document containing all RFD values in 1kb bin along the genome (in bedgraph format)

-SampleName: the name of all input file, multi files will be separated by “,” and the name of bed file can’t contain character “\_”, since it has been used as delimiter in script.

**An Example (Quick start):**

Java -jar Add\_DeltaRFD\_ToSegment.jar

-SampleName 1905.FC0.bed,1905.FC1.bed

-O /Users/wwang/Desktop/ORM/Add\_DeltaRFD\_ToSegment/Test

-I /Volumes/WWT/Final-Version/1905/Bed/sort

-RFD /Users/wwang/Desktop/Final-Version/ForkDirection/AddFDItoBed/RFD\_InWindow/output/1kb/NewRFD\_1kb\_sort.bedgraph

1. Calculate\_FDI\_RFD

Similar to RFD, the FDI\_RFD is also calculated by formula (R-L)/(R+L), but L and R correspond to the numbers of ORM segments with positive and negative FDI values, respectively, overlapping with each 1kb window. Firstly, the segment bed file(s) should be treated by the package Add\_FDI\_ToSegment.jar, then we can map these regions into 1kb adjacent bins along the genome and calculate the corresponding FDI\_RFD values.

-REF: The input REF file for the reference chromosome length, unit nt, see below for an example (can find for example at UCSC genome browser):

chr1 249250621

chr2 243199373

chr3 198022430

…

Please pay attention, here the chromosome names must be the same as the chromosome names show up in .txt files (case sensitive, i.e. Chr and chr is different). If using chr23 and chr24 to represent chrX and chrY, respectively, in .txt file, the corresponding chromosome names in this REF file should be chr23 and chr 24 instead of chrX and chrY.

-WS: window size, unit kb

-I: The parent path for multi bed files with FDI values or a single bed file with FDI values

-O: The parent path for multi results with FDI\_RFD or a single bed file with FDI\_RFD

-Colum\_FDI: The Colum number containing the FDI values within the input file(s)

-SignalColumn: The Colum number of input file’s signal number. If the bed file is generated by jar package GetNewSegmentation\_AddSoloSignal.jar, there should be one column is signal number. If the bed file contains this column, user can set the cutoff for filtering segments with few signals to reduce the false positive. The suggested cutoff is 3. The default value is 2, if not set.

-FDI\_N\_Cutoff: The cutoff value for filtering the segments with negative FDI. The default will be zero if not set.

-FDI\_P\_Cutoff: The cutoff value for filtering the segmentation with positive FDI. The default will be zero if not set.

-Signal\_N: The cutoff value for filtering the segments with negative FDI containing signals lower than the cutoff. The default will be zero if not set.

-Signal\_P: The cutoff value for filtering the segments with positive FDI containing signals lower than the cutoff. The default will be zero if not set.

Notice: Please pay attention -I and -O is direct path instead of parent path and the script can detect the chr23 and chr24 to turn into chrX and chrY. So the corresponding chromosome in -REF path file should be chrX and chrY.

**An Example (Quick start):**

Single bed input with FDI cutoff setting

Java -jar -REF  /Users/wwang/Desktop/enrichment/ChrLength\_hg19.txt

-WS 1

-I /Users/wwang/Desktop/ORM/FDI\_RFD/Test/All0min.bed

-O /Users/wwang/Desktop/ORM/FDI\_RFD/Test/Output.bed

-Colum\_FDI 7

-SignalColumn 5

-SignalCutoff 2

-FDI\_P\_Cutoff 1

-FDI\_N\_Cutoff -1

Multi bed input with default FDI cutoff

Java -jar -REF  /Users/wwang/Desktop/enrichment/ChrLength\_hg19.txt

-WS 1

-I /Users/wwang/Desktop/ORM/FDI\_RFD/Test/

-O /Users/wwang/Desktop/ORM/FDI\_RFD/Test/Output.bed

-Colum\_FDI 7

-SignalColumn 5

-SignalCutoff 2

-SampleName 1905.FC0.bed,1905.FC1.bed

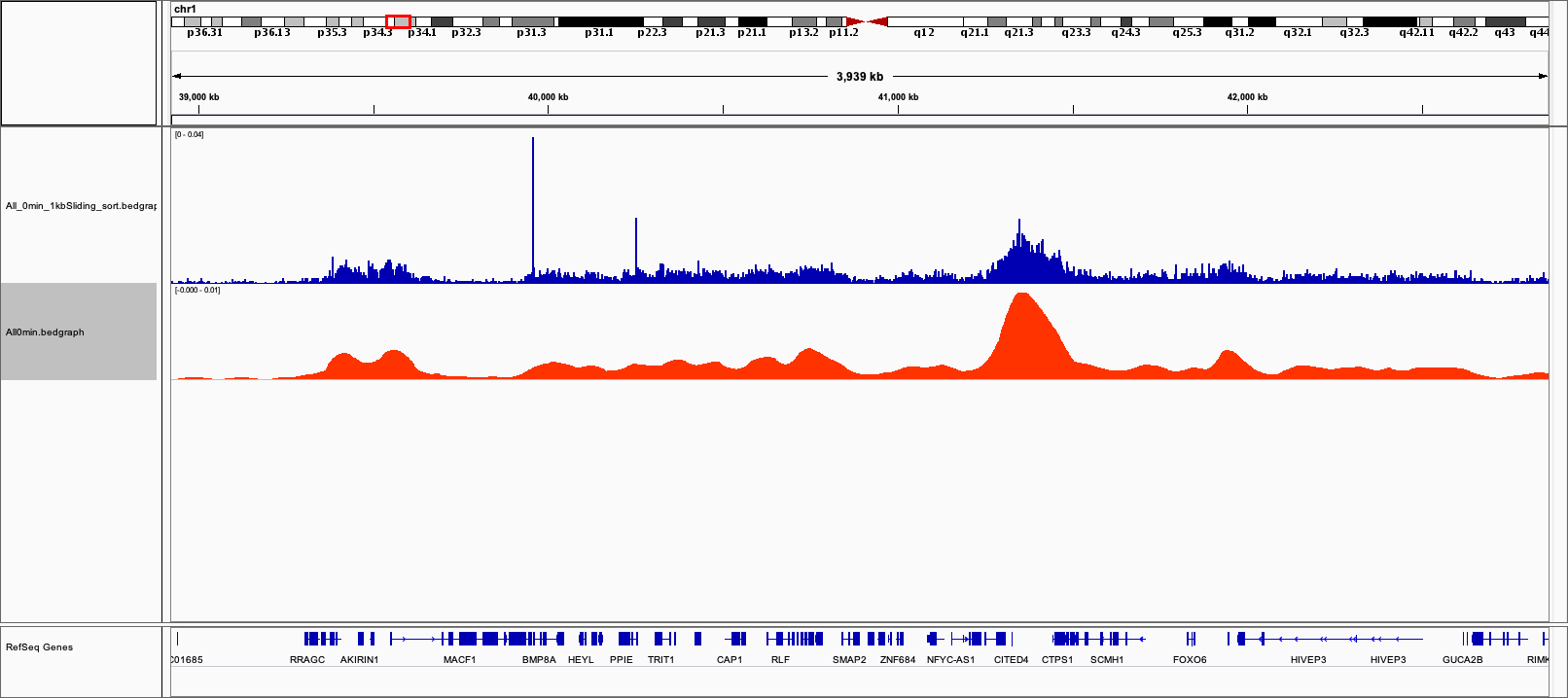
1. LOESS.R  
    The LOESS.R is used to fit the signal fire efficiency in adjacent bins into continuous curve. The input file should have at least 4 columns. The chromosome, start, end and fire efficiency (or other value to smooth). The function Fitcurve() has 4 parameters. The input data, output bin size, transition region length (to eliminate the gap between two adjacent bins) and the signal column for smoothing.

Parameters

1. Input: The data frame of your input file path
2. The length of output bin: Please pay attention the output bin length should be the integer multiple of input bin length, at least same as the input bin size, unit kb.
3. Because the LOESS fitting is impossible to do along the whole chromosome. It will be very time consuming. So, we set a buffer transition area equally divided across 2 adjacent output bins. And just do the LOESS fitting in each output bin and extending half transition area inside the adjacent output bin range, which means within the transition area, all signals are fitted by twice. the final smoothed values are the averages of the smoothed values from two adjacent windows, weighted by their distance to the adjacent window.

**An Example (Quick start):**

Input <- read.table(xxx/Your files’ path)   
Fit <- FitCurve(Input,20,10,4)  
## Annotation: The output bin size is 20kb, transition area length is 10kb, the fire efficiency needs to LOESS smooth is in 4th column



**Comparison between LOESS output signals and raw signals**

1. Multi\_peakcalling.R

After LOESS smoothing fitting by LOESS.R, we can do peak calling for the smoothed fitting signals. There are 2 functions.

Function1: GetInitialZone\_byBin()

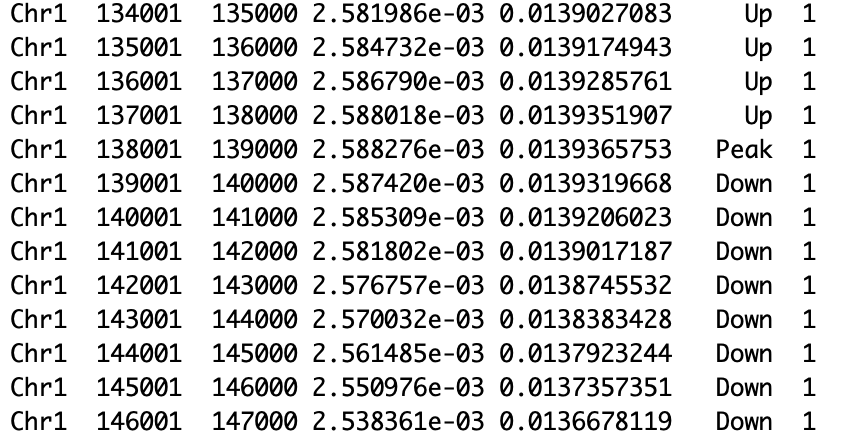
It just needs the LOESS fitting result as input or other continuous fitting signals

An Example (Quick start):

Input <- read.table(xxx/Your LOESS fitting files’ path)   
Peak <- GetInitialZone\_byBin(Input)

**Output format**

**The output file containing 7 columns corresponding to**  
Chr Bin\_start Bin\_end Fitting\_Signal\_value Signal\_Percentage Trend\_Label InitialZone\_Sign



At first, the positions of peaks and the valleys are defined as the local maximum and local minimum, respectively, on the continuous normalized ORM signal density profile. And an initiation zone region is defined from a peak (labeled as Peak at the 6th column) to the closest valleys (labelled as Valley at the 6th column) on both sides. All the bins within such range will be marked as 1 for InitialZone\_Sign column. Then, based on the value of the normalized ORM signal density belong to each bin, if it is bigger or smaller than the adjacent bins, each bin can be labelled as Up or Down, respectively, at the 6th column. Finally, we will calculate the sum of smoothing signal densities in the 4th column within the entire initial zone, and calculate the percentage of each bin’s signal to this sum value reported as signal percentage at the 5th column, which are then used to narrow down to get a more precise definition of an initial zone. Here in our study, based on our investigation, the size of each initiation zones was defined by the smallest window that containing at least 40% of ORM signals corresponding each peak (see our manuscript for more detailed information).

Function2: GetInitialZone()

This function is used to select the initial zones detected in several biological replicates from the outputs obtained by GetInitialZone\_byBin(). It needs 3 parameters, the outputs of GetInitialZone\_byBin(), the vector recoding for each bin in which biological replicate(s) an initiation zone being identified (based on the 7th column), and the label vector for each biological replicates.An Example (Quick start):  
  
S1708 <- read.table("xxx/Your GetInitialZone\_byBin() output")

S1802 <- read.table("xxx/Your GetInitialZone\_byBin() output")

S1807 <- read.table("xxx/Your GetInitialZone\_byBin() output")

S1905 <- read.table("xxx/Your GetInitialZone\_byBin() output ")

IZ <- list(S1708,S1802,S1807,S1905)

Sign <- c("a","b","c","d")

SampleNumber <- rep(0,nrow(IZ[[1]]))

for(i in 1:4)

{

Tmp <- IZ[[i]]

Tmp[,8] <- "0"

Tmp[which(IZ[[i]]$V7 > 0),8] <- Sign[i]

SampleNumber <- paste(SampleNumber,Tmp$V8,sep = "")

}

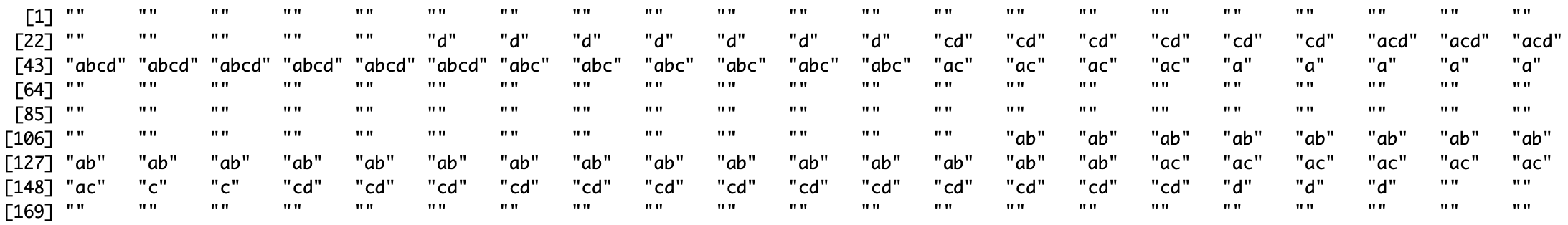
library(stringr)

SN <- str\_replace\_all(SampleNumber, "0", "")

All0min <- read.table("/Users/wwang/Desktop/Final-Version/Final\_Initialzone/IZ\_Update/SignalRatio/Sliding/All0min\_SignalRatio.bed")

FinalIZ <- GetInitialZone(All0min,SN, Sign)

Here we used "a","b","c","d" label for 4 replicates as label vector and SN record all the initial zone show up in each bin like below.



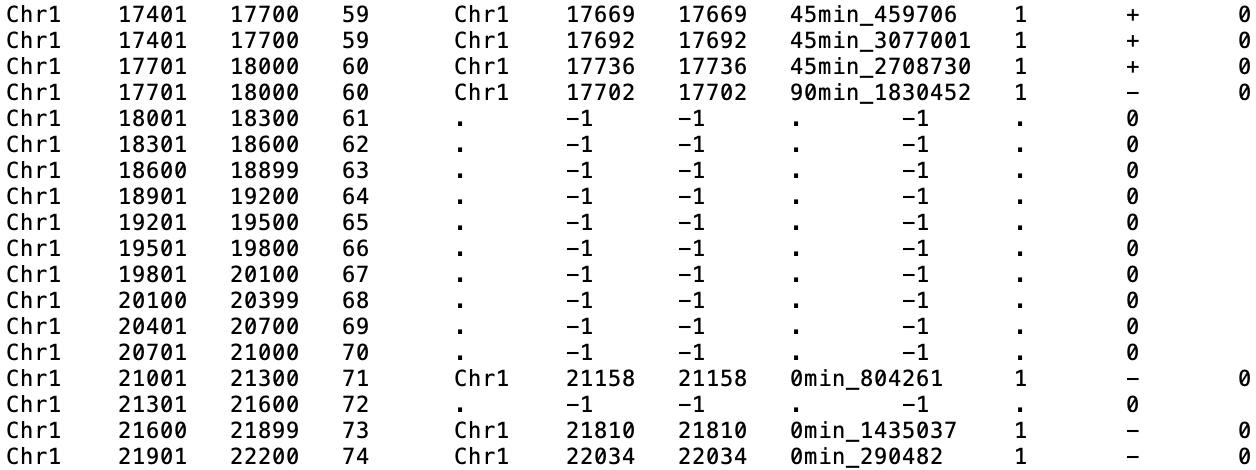
The "" means no replicate has initial zone in this bin and if there are any replicate contain the initial zone it will show up corresponding replicate’s label.

1. GetMappedCount.jar

Sometimes we need to count the segmentation numbers overlapped with specific regions. For example, we have 2 bed files, one is T-peak regions. The other is ORM origin segmentations. We want to know how many ORM segments overlapped with T-peak regions. This time, we need used bedtools to mapped the ORM segments to Tpeak regions. The linux command is like below.

Bedtools -a Tpeak\_Region.bed -b ORM\_Segment.bed -wao > Map.bed

The output content of Map.bed is like below



The line contains “-1” and “.” is the Tpeak regions without any overlapped ORM segments, so their mapped count is 0

But some regions have more than 1 overlapped ORM segments, like regions with ID (4th column) 59 and 60 have 2 overlapped ORM segments, so their mapped count is 2.

And sometimes ORM segment may overlap with two Tpeak regions. This time we can count it in both Tpeak regions or when we have a much stricter requirement for overlapping, we only count the Tpeak regions where the central position of ORM segment located in.

Parameters:

-I: the parent path of the mapped bed file(s) that you got by bedtools

-O: the parent path of the output file(s)

-SampleName: the name of all input file, multi files will be separated by “,” and the name of bed file can’t contain character “\_”, since it has been used as delimiter in script.

-IDColum: The unique ID column for the Tpeak regions in example. The above example the ID column is the 4th column

-MapStartColum: The start column where mapped segments, in example the start column is 6th

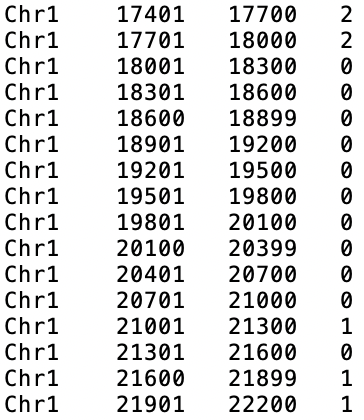
-MapEndColum: The end column where mapped segments, in example the start column is 7th

-MidCount: it could only be “Y”, the default value is “N”, when the item set as “Y”. The script only count the overlapped segment whose central position located inside the mapped regions.

Notice: Besides the -MidCount, all items are must-have items, includes MapStartColum and MapEndColum, they are used to detect if one region without any overlapping segments, and when -MidCount is “Y”, they are active to calculate the central position of segmentation

Output:

The column is chromosome, start, end and mapped number



An Example (Quick start):

Java -jar GetMappedCount.jar

-SampleName Map1.bed,Map2.bed

-O xxx

-I xxx

-IDColum 4

-MapStartColum 6

-MapEndColum 7

1. GetMapped\_All\_info.jar

Similar like GetMappedCount.jar, when mapped files are bedgraph files with specific values. Sometimes we want to get the sum of all mapped value and the average values. We can use this jar package to get all mapped information

In this package, we changed -IDColum, -MapStartColum and -MapEndColum into array it should have same length as SampleName and take “,” as delimiter.

Besides that, adding the -KeepColum to set the output format, it means how many columns start from left to right, you want to keep in output files normally we keep the chr, start, end and ID of target segment. It is also an array separated by “,” with same length of SampleName

-Column\_Value is an array to store the column number of all input interested bedgraph files’ values and take “,” as delimiter.

An Example (Quick start):

Java -jar GetMapped\_All\_info.jar

-I /Volumes/WWT/Final-Version/Cormatrix\_Rawdata/Map/50kb\_bin/MCM/   
-O /Volumes/WWT/Final-Version/Cormatrix\_Rawdata/SUM/50kb\_bin/Mix/MCM/   
-SampleName MCM\_R1\_Map.bed,MCM\_R2\_Map.bed   
-MapStartColum 6,6   
-MapEndColum 7,7   
-IDColum 4,4   
-MidCount Y   
-KeepColum 4,4   
-Column\_Value 8,8

1. Abstract\_HotBin.R

This R package is used to set the proper cutoff value for hot dots and pick out the hot spot regions. (The detailed pipeline could be checked by the link below.

<https://elabftw.curie.fr/experiments.php?mode=view&id=19339>

1. FilterHotDot.jar

When we have the hot spot regions and .TXT file contains the hot spots. We can use this script to pick out all green mapping signals’ and red labeling signals’ intensity to check the signal intensity distribution difference inside and outside the hot spots’ regions. And get the .TXT file after hot spots filtering

Parameters:

-IHR: the parent path of the hot sports regions

- TXT: the parent path of the .txt file contains the hot sports

-SampleName: the name of all input file, multi files will be separated by “,” and the name of bed file can’t contain character “\_”, since it has been used as delimiter in script.

-O: the parent path of the .txt file after filtering the hot sports

-IntensityInRegion: The parent path for the red and green intensity inside the hotspots’ region

-IntensityOutRegion: The parent path for the red and green intensity outside the hotspots’ region

**Output format**   
The output of .txt file has the same format of input .txt file. The only difference is filtering the hot sports.

An Example (Quick start):

Java -jar GetMappedCount.jar

-IHR HotSportsRegion.bed

-SampleName 1802,1807

-O xxx(The parent path for output filtering .txt files)

-TXT xxx(Your path to input TXT contain hot spots)

-IntensityInRegion xxx(The parent path for output signal density inside the hot sports regions)

-IntensityOutRegion xxx(The parent path for output signal density outside the hot sports regions)

1. SignalCompare.R

This R package is used to plot the signal intensity distribution inside and outside the hot sports regions to compare. (The detailed pipeline could be checked by the link below.)

<https://elabftw.curie.fr/experiments.php?mode=view&id=19339>