Table of Contents

Methyl-SNP-seq README

Author: Bo Yan, yan@neb.com
Revision Date: June 7, 2022

Introduction

Methyl-SNP-seq is taking advantage of the double stranded nature of DNA to duplicate the sequence information into a linked copy to the original strand that is resistant to bisulfite conversion. After conversion, the copied strand conserves its original four nucleotide content while the original strand undergoes deamination at unmethylated cytosines. I summarize the principle of Methyl-SNP-seq in the following figure.

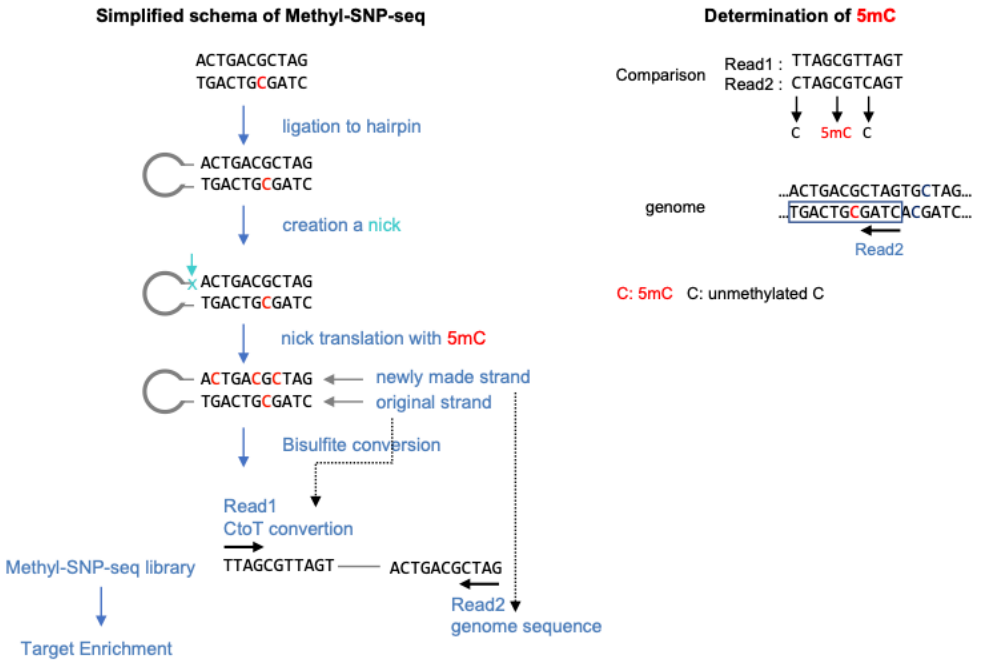

Figure: Principle of Methyl-SNP-seq

The scripts included in the ReadProcessing folder are used for processing the Methyl-SNP-seq sequencing reads including: adapters trim, Read Deconvolution, Removal of duplicates, Removal of multiple mapping, addition of XM tag for methylation extraction.

Functions

Adapter Trimming: TrimRead.py

Synopsis:
TrimRead.py generates and run a bash script containing the commands to trim the illumina adapter as well as the hairpin adapter sequence from the paired end reads.
The trimming of adapter sequence is performed using trim_galore.

Requirement:
Python 2.7
trim_galore (version>= 0.6.4) and cupadapt is preinstalled.

Usage:

```
$python TrimRead.py --Read1 TestSeq.1.fastq.gz --Read2 TestSeq.2.fastq.gz --name TestSeq
```

Parameters:

--Read1, --Read2:
Full path of the illumina sequencing read1, read2 fastq or fastq.gz file, e.g. TestSeq.1.fastq.gz and TestSeq.2.fastq.gz

--name:
Name for output fastq files, e.g. TestSeq
The resulting adapter removed Read1 and Read2 fastq files that are used for next step Read Deconvolution will be: TestSeq_hairpin_R1_val_1.fq and TestSeq_hairpin_R2_val_2.fq
These files are saved under the current working directory.

--path_to_cutadapt, --path_to_trimgalore: Optional
Use this option to specify a path to the cutadapt or trim_galore executable,
e.g. /mnt/home/yan/exe/TrimGalore-0.6.4/trim_galore, /mnt/home/yan/exe/cutadapt
Else by default it is assumed that cutadapt or trim_galore is executable in the PATH.

Reference-free Read Deconvolution: DeconvolutionConversion_v2.py

Synopsis:
DeconvolutionConversion_v2.py performs reference-free Read Deconvolution of Methyl-SNP-seq paired end Read1 and Read2, e.g. for microbiome Methyl-SNP-seq.
Read Deconvolution compares the same sequencing cycle of a paired read (base is referred as R1 and R2), including the following actions:
(1) Base determination and methylation extraction. For the same illumina cycle, if Read1 base is a C and Read2 base is a C, it results in a C in the deconvoluted read and a 5mC in the methylation report; while if Read1 base is a T and Read2 base is a C, it results in a C in the deconvoluted read and a unmethylated C in the methylation report.
(2) Base quality score adjustment. For the mismatching positions that Read1 bases are different from Read2 bases except for the Read1-T Read2-C case, Read1 bases are used but the sequencing quality scores are adjusted to 0 in the deconvoluted reads.
This Deconvolution step generate a fastq file containing deconvoluted reads and a methylation report containing the methylation status of each C in each deconvolution read.

Requirement: Python 2.7

Usage:

```
$python DeconvolutionConversion_v2.py --Read1 TestSeq_hairpin_R1_val_1.fq --Read2
TestSeq_hairpin_R2_val_2.fq --name TestSeq
```

Parameters:

--Read1, --Read2:
Adapter removed Read1 and Read2 fastq files.

--name:
Name for output files:
The resulting Deconvoluted Read fastq file will be e.g. TestSeq_DeconvolutedRead.fq.
The resulting methylation report will be e.g. TestSeq.Deconvolution.5mC.
The methylation report is 0-coordination, e.g.

```
A00336:A00336:HV7F7DRXX:1:1101:10004:10927      C0C2C11C13C20C21C22C25C41C45C51C52C53C60M63C70M71C74
```

First column is the readID, second column marks the methylation status of C in the deconvoluted read, C for unmethylated cytosine and M for 5mC.
These files are saved under the current working directory.

Reference-dependent Read Deconvolution: DeconvolutionWithCalibration package

Synopsis:
This package performs reference-dependent Read Deconvolution of Methyl-SNP-seq paired end Read1 and Read2, e.g. for human Methyl-SNP-seq.
The step (1) Base determination and methylation extraction is the same as the Reference-free Read Deconvolution. But Reference-dependent Read Deconvolution uses a statistical model for the base quality score adjustment as shown below.
(2) Base quality score adjustment. For the mismatching positions, by comparing to the reference genome, a Bayesian probability is calculated, which reflects the likelihood of being able to trust the Read1 base. Therefore, Read1 bases are used but the sequencing quality scores are adjusted based on the Bayesian probability in the deconvoluted reads.

Requirement:
Python 2.7
Bowtie2 is preinstalled.
Bash shell is enabled and executable in the PATH: #!/bin/bash

Usage:
To perform Reference-dependent Read Deconvolution, download the DeconvolutionWithCalibration directory which contains __main__.py and src folder.
Then run the command as follows:

```
$python DeconvolutionWithCalibration --Read1 TestSeq_hairpin_R1_val_1.fq --Read2
TestSeq_hairpin_R2_val_2.fq --name TestSeq --reference hg38.fa
```

Parameters:

--Read1, --Read2:
Adapter removed Read1 and Read2 fastq files. Do not take compressed gz file.

--name:
Name for output files, e.g. TestSeq
TestSeq_DeconvolutedRead.fq: deconvoluted reads
TestSeq.Deconvolution.5mC: methylation report
TestSeq.BaseCalibration.table: base calibration table for Bayesian modeling
TestSeq.BaseCalibration.probability: Bayesian probability table

--reference: Required
Reference genome (.fa) used to map the reads for base calibration.

--percent: float, Default 0.05
Subsample input reads for base calibration.

--vcf: Optional
A vcf file containing known SNP positions. If provided, the positions listed in this vcf file are ignored from base calibration.

--smp: Optional, Default 1
Number of threads used for bowtie2 mapping. Need to assign enough memory to run multiple threads.

--dir: Optional
Directory (full path) to save the output files.

If not provided, the output files are saved at current working directory.

--path_to_python, --path_to_bowtie2: Optional
Use this option to specify a path to the python2.7 and bowtie2 executable, e.g. /usr/bin/python and /usr/bin/bowtie2
Else by default it is assumed that python2.7 and bowtie2 is executable in the PATH.

Read Deconvolution Correction: DeconvolutionUnmatchCorrect.py

Synopsis:
This script is used to correct some of the unmatched read pairs that Read1 and Read2 bases are not aligned properly.
The output fastq files containing the corrected read pairs can be deconvoluted using DeconvolutionWithCalibration or DeconvolutionConversion_v2.py as explained above.

Note:
The selection of -F 4 and -F 256 will be performed by this script, so do not need to apply this selection on the Unmatched Read1 and Unmatched Read2 bam/sam files. The Unmatched bam files do not need to be sorted by coordination.

Requirement:
Python 2.7
samtools preinstalled.

Usage:

```
$python DeconvolutionUnmatchCorrect.py --Read1 TestSeqUnmatch_R1.bam --Read2
TestSeqUnmatch_R2.bam --name TestSeq
```

Parameters:

--Read1: alignment of unmatched Read1 bam/sam file
Unmatched Read1 bam/sam file:
The Read1 fastq file, which contains the reads that cannot be matched using DeconvolutionWithCalibration or DeconvolutionConversion_v2.py in other words not included in the output deconvoluted reads, is aligned using bismark mapping.
The generated bam/sam file is used as Unmatched Read1 bam/sam file.

--Read2: alignment of unmatched Read2 bam/sam file
Unmatched Read2 bam/sam file:
The Read2 fastq file, which contains the reads that cannot be matched using DeconvolutionWithCalibration or DeconvolutionConversion_v2.py in other words not included in the output deconvoluted reads, is aligned using bowtie2 mapping.
The generated bam/sam file is used as Unmatched Read1 bam/sam file.

--name:
Name for output Read1 and Read2 files having the corrected matched reads,
e.g., TestSeq_unmatchcorrect.R1.fastq and TestSeq_unmatchcorrect.R2.fastq.
The output files can be used as input files for Read Deconvolution as shown above.

--pathSamtools: Optional
Use this option to specify a path to the samtools executable,
e.g. /usr/bin/samtools.
Else by default it is assumed that samtools and bedtools is in the PATH.

--dir: Optional
Directory (full path) to save the output files.
If not provided, the output files are saved at current working directory.

Removal of multiple mapping reads: MarkUniread.py

Synopsis:
MarkUniread.py removes the multiple mapping based on bowtie2 mapping. Only the reliable (uniquely) mapping is saved in the output sam file.
Reliable mapping is defined as:
(1) Flag != 4, 256, 2048
(2) XS tag not present or AS tag != XS tag. See bowtie2 manual for definition of XS and AS tag.

Requirement: Python 2.7

Usage:

```
$python MarkUniread.py --input TestSeq_DeconvolutedRead.sam --output
TestSeq_DeconvolutedRead.uni.sam
```

Parameters:

--input:
A sam file sorted by coordination containing the deconvoluted reads.

--output:
A sam file sorted by coordination containing the deconvoluted reads with removal of multiple mapping.

Removal of duplicated reads: MarkDup.py

Synopsis:
MarkDup.py removes the duplicated reads for Single end mapping, e.g. Deconvoluted Read mapping. Here duplicated reads are defined as single end reads mapped to the same chr and locus and have the same sequence in col 10. For duplicates, save one copy with the highest MAPQ score in the output sam file.

Note:
Do not perform -F 256 or -F 1024 or -F 2048 selection in this step. Apply these filters in advance if necessary.

Requirement: Python 2.7

Usage:

```
$python MarkDup.py --input TestSeq_DeconvolutedRead.uni.sam --output
TestSeq_DeconvolutedRead.uni.nodup.sam
```

Parameters:

--input:
A sam file sorted by coordination containing the deconvoluted reads.

--output:
A sam file sorted by coordination containing the deconvoluted reads with removal of duplicates.

Addition of XM tag: AddXMtag.py

Synopsis:
AddXMtag.py adds a XM tag to each mapping in the sam file of the Deconvoluted Read. XM tag is defined by bismark to labeling the methylation status. Therefore, the methylation status at each position could be further extracted using bismark_methylation_extractor.
The determination of methylation status of the deconvoluted reads is similar to the bismark principle as shown below.

```
To report XM tag based on the deconvolution report:
    nonC: position is not a C or Methylated C

    If a position is a C (or Methylated C) in the deconvolution report, and also C on genome:
        Find the methylation context based on the seq in genome, use x,h,z,u or Capital in XM tag following bismark rules.
    If a position is nonC in either deconvolution report or genome:
        Use '.' in XM tag in the corresponding position in SEQ
    If a position is a C on genome but is nonC in deconvolution report:
        Use '.' in XM tag.
    If a position is C in deconvolution report, but nonC on genome (mismatch) or this base is a insertion (I in cigar) not present in genome:
        Use '.' in XM tag.

    The C context is determined based on the corresponding genome, in other words based on the mapping strand from 5' to 3' direction:
        For read mapping to reverse strand (Flag 16),
        e.g. Read in fastq 5'-GcGT-3', SEQ=ACgC (corresponding to top strand 5'->3' direction), bottom genome=5'-GcGT-3', top genome='5-ACgC-3',
        the context is cG depending on the bottom genome .z.., so the XM tag corresponding to SEQ in bam file is reverse of '.z..' -> '..z.'

    So for C at mismatching/SNP site or before mismatching/SNP site, the C context calling could be not accurate.

There is Difference between AddXMtag.py and bismark for calling C coming in front of a deletion (^):
AddXMtag.py decides the context based on the genome context:
        col10:      TTTC^^AAATTATTTTGTGATGTGTGTGTTTAATTTATAGAGTTTAATTTTTTTTTTTTATAGGGTAGTTTGGAAATAT
        my XM:      .h.X  ...h..h.............z...h..h.h.x.........hh...h....h......x..........h.h
    top genome:   attctcagaaactactttgtgatgtgtgcgttcaactcacagagtttaacctttctttcatagggcagtttggaaacactc
bismark labels the C in front of deletion depending on the context of sequence (CAA -> H):
        col10:      TTTC^^AAATTATTTTGTGATGTGTGTGTTTAATTTATAGAGTTTAATTTTTTTTTTTTATAGGGTAGTTTGGAAATAT
    bismark XM:     .h.H  ...h..h.............z...h..h.h.x.........hh...h....h......x..........h.h
    top genome:   attctcagaaactactttgtgatgtgtgcgttcaactcacagagtttaacctttctttcatagggcagtttggaaacactc

No Difference between AddXMtag.py and bismark for calling C coming in front of an insertion (-):
AddXMtag.py decides the context based on the genome context:
        col10:      TTTC-TTTTTTTTGGGCTTTAGTTTCTTTTTTGGTAAAACGGGGATGGTAATGGGATATTCTTAGGGGGTGTATGA
        my XM:      ...H.....hhx....Hh.x.....Hh.h..........Z.................h.hhH.x........h....
    top genome:   gatttc atttccctgggcctcagtttcctctttggtaaaacggggatggtaatgggacaccctcagggggtgcatgagg
bismark labels the C in front of insertion based on the genome context (CAT -> H):
        col10:      TTTC-TTTTTTTTGGGCTTTAGTTTCTTTTTTGGTAAAACGGGGATGGTAATGGGATATTCTTAGGGGGTGTATGA
    bismark XM:     ...H.....hhx....Hh.x.....Hh.h..........Z.................h.hhH.x........h....
    top genome:   gatttc atttccctgggcctcagtttcctctttggtaaaacggggatggtaatgggacaccctcagggggtgcatgagg

No Difference between AddXMtag.py and bismark for calling C at or coming in front of mismatch.

Bismark tags needed for bismark extractor:
    XM-tag (methylation call string)
    XR-tag (read conversion state for the alignment)
    XG-tag (genome conversion state for the alignment)
```

Note:
(1) Creat XM tag for methylation status corresponding to the position in SEQ
(2) The suspending part in SEQ (S in cigar) is not shown in XM tag, so len(XM) = sum of cigar M/I
(3) Do not consider the scenario that cigar has hard clip H.
(4) Deconvolution report may contain entries without methylation information (col2 is empty). In this case, report XM tag using '.' for all the positions.
(5) AddXMtag.py performs -F 4 to remove the unmapped reads, but not apply other Flag tag filter.
(6) Add --thread to run multiple threads (pathos node=4) to speed up for a large sam file. In this case, the input sam file is split into sub sam files having 10 million alignments each. The size of a sub sam file having 10 million alignments is 4G for 150bp sequencing, 2.6G for 100bp

sequencing. Each sub sam file requires about 7-8G memory. Make sure to assign enough memory and space to run multiple threads.
(7) The reads in output sam file are in the same order as in the input sam file.

Requirement:
Python 2.7, pathos modules installed for multiple threads
bedtools and samtools preinstalled.

Usage:

```
$ python AddXMtag.py --input TestSeq_DeconvolutedRead.uni.nodup.sam  --report
TestSeq.Deconvolution.5mC --name TestSeq --output_dir /Users/tmp --reference hg38.fa --
path_to_samtools /Users/yan/miniconda3/envs/my-bowtie2-env/bin/samtools --path_to_bedtools
/Users/yan/miniconda3/envs/my-bowtie2-env/bin/bedtools --thread
```

Parameters:

--input:
A sam file that is generated by aligning the Deconvoluted Read to the reference genome.

--report:
A methylation report generated during Deconvolution Step.

--name:
Name for output files, e.g.
name.XMtag.sam: saving all the entries with XMtag added;
name.noXMtag.sam: saving all the entries that cannot be added XMtag: unmapped reads, reads without information in deconvolution report or reads that cannot be analyzed for methylation context (at the boundary of chr).

--reference:
Reference genome (.fa) used to align the deconvoluted reads.

--dir: Optional
Dir for saving output files, not required.
If not provided, output files are saved in the current working dir.

--thread: Optional
Add --thread to run multiple threads using pathos if input Read1.sam is a large file.
Need to assign enough space and memory. See note (6).

--pathSamtools, --pathBedtools: Optional
Use this option to specify a path to the samtools or bedtools executable,
e.g. /usr/bin/samtools or /usr/bin/bedtools.
Else by default it is assumed that samtools and bedtools is in the PATH.