SAM filtering pipeline (SFP)

Sofie O'Brien and Wei-Shou Hu
Department of Chemical Engineering and Materials Science, University of Minnesota

Manual Contents

1. Program Description

SFP is a Python script for identifying integration site(s) of a transfected vector from next generation genome sequencing data. It is best suited for targeted sequence data, such as that from sequence capture of probed vector regions. However, it will also work for whole genome sequencing data, though the memory requirements are large (the more reads in your data set, the larger the memory requirements). A bwa-mem mapped .sam file is required as input to the pipeline.

2. Citation

If you use SFP, please cite the following paper and the deposited software:

O'Brien, S.A., Ojha, J., Wu, P. and Hu, W.-S. (2020), Multiplexed clonality verification of cell lines for protein biologic production. *Biotechnology Progress*, e2978. https://doi.org/10.1002/btpr.2978

O'Brien, Sofie A; Hu, Wei-Shou. (2019). SAM Filtering Pipeline (SFP): Algorithm for the determination of integration sites from next generation sequencing data. Retrieved from the Data Repository for the University of Minnesota, https://doi.org/10.13020/9wgm-mj51.

3. Files in repository

a) "SAM_filtering_pipeline.py": Python script needed to run integration site identification pipeline from mapped .sam files
b) "Manual.pdf": This file.

4. Required software

a) Pre-processing of sequencing data

The following software was used to pre-process the sequencing data before running SFP:

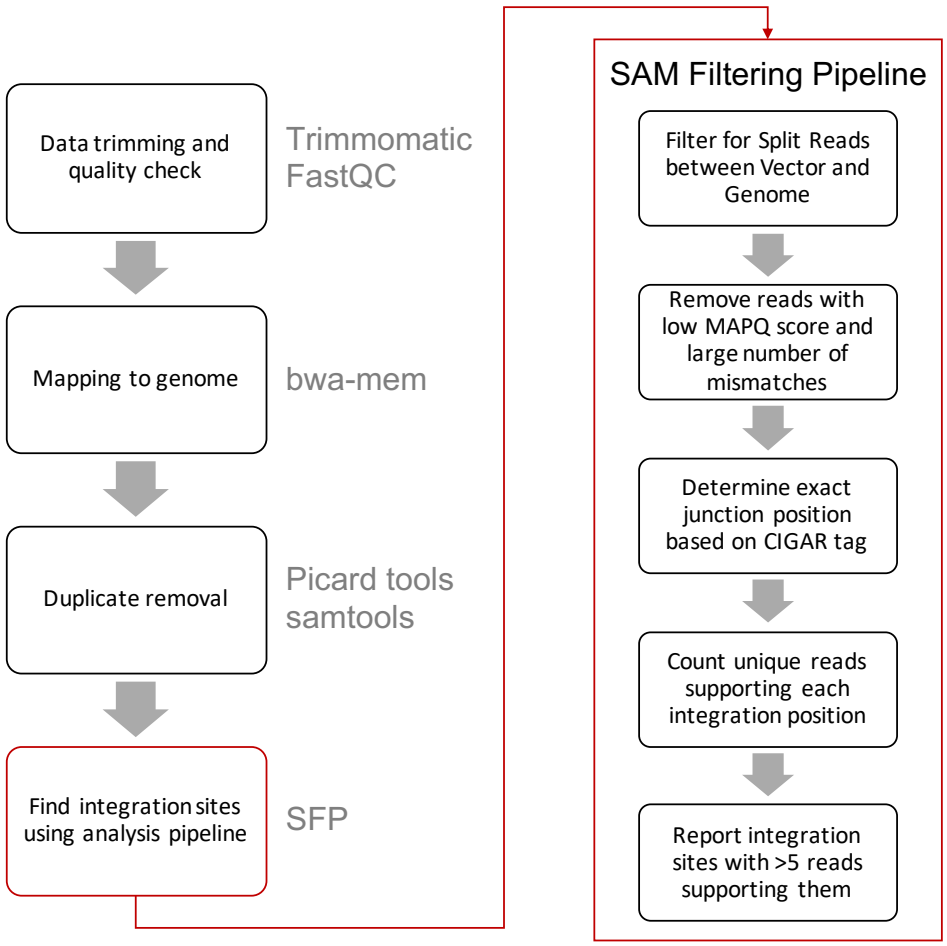
- Trimmomatic 0.33
- FastQC
- bwa 0.7.17
- picard-tools 2.18.16
- samtools 1.9

b) Running the SAM Filtering Pipeline (SFP)

To run the algorithm, Python 3.6.3 or newer is required. Additionally, the following Python libraries are needed:

- mmap
- re
- numpy
- pandas
- sys
- os
- argparse

5. Overall data analysis workflow:



6. Preparing genome for mapping

SFP looks for reads which are split between a transfected vector and the genome of the cell line used. For this reason, the vector sequence must be added to your genome file before mapping sequencing reads. This can be done with a concatenate command on unix systems to merge the genome and vector fasta files:

```
cat Genome.fa Vector.fa > Genome_with_Vector.fa
```

Note: Make sure to give the Vector scaffold a unique name. One of the required inputs for SFP is the name of the Vector scaffold.

7. Pre-processing of data

   SFP takes a bwa-mem mapped .sam file as input. Since the algorithm uses specific tags generated by bwa-mem to identify split reads, it is not compatible with other mapping algorithms. The following is a recommended list of pre-processing steps before running SFP. Other software may be used for steps other than mapping.

   a) Trimming of FASTQ files from sequencing using Trimmomatic and extracting trimmed and paired reads if using paired-end data.

   b) Checking quality of sequencing using FastQC to ensure trimming removed any adapter sequences and low-quality bases.

   c) Mapping reads to genome containing vector sequence as an added scaffold using bwa-mem.
      i. bwa-mem requires the genome to be indexed by bwa before mapping.

   d) Removal of PCR duplicates from mapped reads using picard tools to prevent over-representation of reads supporting a particular integration site.
      i. This is especially recommended for sequencing preparations which use multiple PCR steps to amplify the library, such as those from targeted sequence capture.
      ii. Without duplicate removal, the minimum number of reads needed to support a "true" integration site may need to be adjusted, depending on the method used to generate the sequencing data.

   At the end of these steps, a duplicate removed .sam file mapped by bwa-mem should be obtained. This file will be used as input for SFP.

8. Running the SAM Filtering Pipeline (SFP)

   To run SFP, call the script from Python:

   python SAM_filtering_pipeline.py -s mapped_data.sam -v Vector

   The program has the following options:

-h	Open the help file
-s mapped_data.sam	Input file name **(required input)**. Name of .sam file will be used as sample name for output files.
-v Vector	Name of vector scaffold (see step 5) **(required input)**
-c integer	Minimum number of split reads required to call an integration site. Default is 5. Entering 0 will use a cutoff of greater than the mean + 1 standard deviation of reads per integration junction found.
-p 1 or -p 0	Whether to run the script in paired end (1, default) or single end (0) mode. Can change the mode to single

	end if using merged paired end reads or single end data.
-i /home/folder/	Directory containing the input sam file. Default is your current working directory.
-o /home/folder/results	Directory to output the results into. Default is your current working directory
-m integer	MAPQ score cutoff for filtering split reads. 30 is the default, can increase to increase stringency of quality. 60 is the maximum possible score for bwa-mem mapped files.
-n integer	Maximum number of mismatches for split reads. 3 is the default, can lower this number to only keep reads with fewer mismatches to the genome.

9. Pipeline output

   The pipeline outputs the following files:

   a) Files generated for each sample (example file names are based on running SFP on mapped_data.sam):

mapped_data_SR.sam SAM file with only split reads between vector and genome retained
mapped_data_SR_readjunctions.txt Table describing the vector-genome junction for each split read. Columns: 1. ReadName – ID for the read 2. PairEnd - R1 or R2 if data is paired end 3. Scaffold – genome scaffold for junction 4. ScaffoldPosition – position on genome scaffold at the junction 5. Vector – Name of vector scaffold 6. VectorPosition - vector scaffold position at the junction
mapped_data_IntSites_all.txt Table of all integration sites found for a given sample, and the number of split reads supporting each site. Columns: 1. Scaffold – genome scaffold for junction 2. ScaffoldPosition – position on genome scaffold at the junction 3. Vector – Name of vector scaffold 4. VectorPosition - vector scaffold position at the junction 5. ReadCount – number of reads supporting the vector-genome junction

6. UniqueReadCount (for paired-end data only) – number of unique read pairs supporting each integration site

mapped_data_IntSites_filtered.txt

Table of integration sites which passed the user defined cutoff for number of split-reads supporting them. UniqueReadCount is used for paired-end data, ReadCount is used for single-end data. Columns:

1. Scaffold – genome scaffold for junction
2. ScaffoldPosition – position on genome scaffold at the junction
3. Vector – Name of vector scaffold
4. VectorPosition - vector scaffold position at the junction
5. ReadCount – number of reads supporting the vector-genome junction
6. UniqueReadCount (for paired-end data only) – number of unique read pairs supporting each integration site

b) Summary files generated for all samples run using the same output directory:

Allsamples_filtered_integration_sites.txt

Table of integration sites which passed the cutoff for number of split-reads supporting them for each sample. If running SFP on multiple samples while using the same output directory, new samples will be added to the existing file. Columns:

1. Sample – sample name
2. Scaffold – genome scaffold for junction
3. ScaffoldPosition – position on genome scaffold at the junction
4. Vector – Name of vector scaffold
5. VectorPosition - vector scaffold position at the junction
6. ReadCount – number of reads supporting the vector-genome junction
7. UniqueReadCount (for paired-end data only) – number of unique read pairs supporting each integration site

Sorted_reads_statistics.txt (for paired end data)

Table describing number of integration sites for each sample, and the percent of reads mapped entirely to genome, entirely to vector, and split reads. If running SFP on multiple samples while using the same output directory, new samples will be added to the existing file. Columns:

1. Sample – sample name
2. IntSite# – number of integration sites found
3. TotalR1 – total reads for R1 of paired end reads
4. TotalR2 – total reads for R2 of paired end reads
5. VectorOnlyR1 – number of reads mapping entirely to vector for R1
6. VectorOnlyR2 – number of reads mapping entirely to vector for R2
7. GenomeOnlyR1 – number of reads mapping entirely to genome for R1
8. GenomeOnlyR2 – number of reads mapping entirely to genome for R2

9. SR_Vector-GenomeR1 – number of split reads between vector and genome that passed quality for R1
10. SR_Vector-GenomeR2 – number of split reads between vector and genome that passed quality for R2
11. SR_lowqualR1 – number of split reads between vector and genome that failed quality for R1
12. SR_lowqualR2 – number of split reads between vector and genome that failed quality for R2
13. UnmappedR1 – number of unmapped reads for R1
14. UnmappedR2 – number of unmapped reads for R2
15. MeanReads/Site – for all of the integration site that passed the cutoff, the average number of reads ± standard deviation that supported each site
16. ReadCutoff – read cutoff used to call a vector-genome junction an integration site

Sorted_reads_unique_statistics.txt (for paired end data)

Table of statistics which counts each read pair (same read id, but R1 and R2) as only 1. Used to identify number of unique DNA fragments that fall under each category of reads (mapped to genome, vector, split reads, and unmapped). Columns:

1. Sample – sample name
2. TotalReads – total number of read pairs
3. VectorOnly – number of reads mapping entirely to vector
4. GenomeOnly – number of reads mapping entirely to genome
5. SR_Vector-Genome – number of split reads between vector and genome that passed quality
6. SR_lowqual – number of split reads between vector and genome that failed quality
7. Unmapped – number of unmapped reads

Sorted_SEreads_statistics.txt (for single end data)

Table describing number of integration sites for each sample, and the percent of reads mapped entirely to genome, entirely to vector, and split reads. If running SFP on multiple samples while using the same output directory, new samples will be added to the existing file. Columns:

1. Sample – sample name
2. IntSite# – number of integration sites found
3. TotalReads – total number of reads
4. VectorOnly – number of reads mapping entirely to vector
5. GenomeOnly – number of reads mapping entirely to genome
6. SR_Vector-Genome – number of split reads between vector and genome that passed quality
7. SR_lowqual – number of split reads between vector and genome that failed quality
8. Unmapped – number of unmapped reads

9. MeanReads/Site – for all of the integration site that passed the cutoff, the average number of reads ± standard deviation that supported each site
10. ReadCutoff – read cutoff used to call a vector-genome junction an integration site