Data analysis procedure for SMURF-seq reads

Contents

1 Introduction

SMURF-seq is a protocol to efficiently sequence short DNA molecules on a long-read sequencer by randomly ligating them to form long molecules. The SMURF-seq protocol involves cleaving the genomic DNA into short fragments. These fragmented molecules are then randomly ligated back together to form artificial, long DNA molecules. The long re-ligated molecules are sequenced following the standard MinION library preparation protocol. After (or possibly concurrent with) sequencing, the SMURF-seq reads are mapped to the reference genome in a way that simultaneously splits them into their constituent fragments, each aligning to a distinct location in the genome (for most fragments).

This manual explains how to map SMURF-seq reads, generate copy-number profiles from the mapped fragments, and perform additional optional analysis of the sequenced read or the mapped fragments.

2 Mapping SMURF-seq reads

At this point, we assume that the reads generated from a SMURF-seq experiment are base-called and are in a fastq or fasta file.

The reads sequenced using SMURF-seq protocol needs to be mapped to the reference genome to identify the fragment locations. The reads can be aligned leveraging long-read mapping tools that are designed for split-read alignment.

We present the option of aligning SMURF-seq reads using BWA-MEM [1], Minimap2 [2], and LAST [3], and we present several parameter recommendations for each tool.

2.1 Prerequisites

Reference genome: The CNV analysis procedure described in section 3 makes use of the human reference genome build `hg19`, and thus this build has to used to generate CNV profiles using the procedure described here. However, other reference genomes can be utilized if the user does not used the procedure in 3

Software required:

1. Mapping tool: One of BWA, Minimap2, or LAST.
2. samtools [4] (version: 1.9)

Environment variables: Required only when using the provided scripts for mapping SMURF-seq reads. The variable `MAPPER` can be set using:

```
$ export MAPPER=<path to mapping tool>
```

2.2 Mapping SMURF-seq reads to the reference genome

A user has an option of using either of the tools listed above following the procedure in section 2.2.1, 2.2.2, or 2.2.3 respectively. We recommend using BWA-MEM as this produced higher fragment counts.

2.2.1 Mapping SMURF-seq reads with BWA-MEM

The download and install instructions for BWA can be found at: `https://github.com/lh3/bwa`

Reference genome index creation: The reference genome index is created using the command:

```
$ bwa index <genome file>
```

The index occupies approximately 5.14 GB for hg19.

Parameter recommendations: The default parameter to map nanopore reads using BWA-MEM is `-x ont2d`. SMURF-seq reads can be aligned using just this option, however, the fragment lengths were longer than optimal. We recommend using the parameters `-A 1 -B 2 -O 0 -E 2` in addition to `-x ont2d`. These parameters constrain the growth of a fragment and their lengths were close to optimal. To further increase the number of fragments obtained, at the expense of a longer runtime, the minimum seed length (`k`) and the minimum chain weight (`W`) can be lowered.

Aligning SMURF-seq reads: SMURF-seq reads are mapped to the reference genome using for fast mapping:

```
$ bwa mem -x ont2d -A 1 -B 2 -O 0 -E 2 -t <threads> \
     <index> <reads>
```

or using to obtain a higher fragment count:

```
$ bwa mem -x ont2d -k 11 -W 5 -A 1 -B 2 -O 0 -E 2 -t <threads> \
     <index> <reads> > <outfile>.sam
```

The above commands can also be run using the scripts:

```
$ ./map/smurfseq_BWA_fast.sh <index> <reads>
```

or

```
$ ./map/smurfseq_BWA_frags.sh <index> <reads>
```

respectively. These scripts require environment variable `MAPPER` set to the location of BWA.

2.2.2 Mapping SMURF-seq reads with Minimap2

The download and install instructions for Minimap2 can be found at: `https://github.com/lh3/minimap2`

Reference genome index creation (Optional): Minimap2 can create minimizer index for the human reference genome in a few minutes before mapping the reads. Optionally, the index can be pre-built and saved to save time during mapping with the command:

```
$ minimap2 -d -w 1 <index name>.mmi <genome file>
```

The index occupies approximately 28GB for hg19. Note that index size is much larger than when using the default parameter. As explained below, this option produces a significantly higher fragment count.

Parameter recommendations: Minmap2 produces higher fragment counts when the window size (`w`) is lowered to 1 and the chain weight (`m`) to 10 than using the default parameters. However, lowering the window size increases the genome index size.

Aligning SMURF-seq reads: SMURF-seq reads are aligned to the reference genome using:

```
$ minimap2 -a -w 1 -m 10 -t <threads> <genome> <reads> > \
     <outfile>.sam
```

or using the script:

```
$ ./map/smurfseq_minimap2.sh <genome file> <reads>
```

This script requires environment variable `MAPPER` set to the location of Minimap2.

2.2.3 Mapping SMURF-seq reads with LAST

The download and install instructions for LAST can be found at: `http://last.cbrc.jp/`

Reference genome index creation: The reference genome index is created using the command:

```
$ lastdb -uNEAR -R01 <index> <genome>
```

The index occupies approximately 15.3GB for hg19.

Parameter recommendations: The genome index is created using the seed patter NEAR (1111110) which is recommended for scheme for finding short-and-strong matches (`http://last.cbrc.jp/doc/last-seeds.html`). The reads are then aligned to the reference genome using the default parameters.

Aligning SMURF-seq reads: SMURF-seq reads are aligned to the reference genome using:

```
$ lastal -Q0 -P<threads> <index> <reads> | last-split | \
     maf-convert sam > <outfile>.sam
```

or using the script:

```
$ ./map/smurfseq_last.sh <index> <reads>
```

This script requires environment variable `MAPPER` set to the location of LAST.

2.3 Generating mapping statistics

After the reads are aligned, the number of fragments generated can be determined using:

```
$ samtools flagstat <mapped file>.sam
```

The number corresponding to the line "mapped" in the output of the above command is the number of fragments generated mapping. Refer to section 4 for optional additional analysis of mapped fragments.

2.4 Mapping subsets of reads in parallel

2.5 Test data

3 Generation of copy-number profiles

CNV profiles were generated using the procedure described in [5, 6] with the modification employed in [7, 8]. Briefly, the human reference genome (hg19) was split into 5k (20k or 50k) bins containing the an equal number of uniquely mappable locations and the bin counts were determined using uniquely mapped fragments. Bins with spuriously high counts ('bad bins', typically around centromeric and telomeric regions) were masked for downstream analysis [6]. This procedure normalizes bin counts for biases correlated with GC content by fitting a LOWESS curve to the GC content by bin count, and subtracting the LOWESS estimate from each bin [6]. Circular binary segmentation (CBS) [9], implemented in DNAcopy [10] package, then identifies breakpoints in the normalized bin counts. Following [7, 8], after CBS, spurious segmentation calls were removed.

3.1 Prerequisites

Software required:

1. python
2. R

Libraries required:

1. DNAcopy (R package) [10]

3.2 Generating CNV profiles

At this point, we assume that the sequenced SMURF-seq reads are mapped to the reference genome and the alignments produced are in a sam file. Generating CNV profiles from a sam file involves getting the uniquely aligned fragment, generating the number of fragments that are aligned to each bin, and generating the segmented CNV profiles using DNAcopy. These step are described below.

Getting uniquely aligned fragments After alignment, some fragments may be ambiguously aligned, i.e. they are equally likely aligned to two or more locations on the reference genome. Since the true location of such a fragment cannot be determined they are removed from copy number analysis. All fragments aligned with a mapping quality (MAPQ in the sam file) greater than 0 are considered as uniquely aligned for copy number analysis. Note that different mappers use different metrics to determine MAPQ. BWA-MEM sets MAPQ to 0 if there are multiple alignments with the same alignment score or if an alignment has a low number of matches and the alignment score of the second best alignment is close to the best alignment.

Uniquely aligned fragments can be generated using the command:

```
$ ./cnv/get_unique_maps.py <aligned>.sam > <alinged>.unique.sam
```

Generating bin counts The next step in CNV analysis is to determine the number of fragments aligned to each bin on the reference genome. This is done with the command:

```
$ ./cnv/get_bin_counts.py <alinged>.unique.sam <chrom sizes> \
     <bin boundaries> <sample name>.bincount.txt <sample name>.stats.txt
```

The files `<chrom sizes>` and `<bin boundaries>` contain the length of hg19 chromosomes and the bin boundaries to determined the counts respectively. These files are located in the the folders cnv/5k, cnv/20k and cnv/50k for 5k, 20k, and 50k bin resolutions respectively.

Generating copy number profiles Finally, the CNV profiles are generated by segmenting the ratio to mean of the bin counts. The profile can be obtained using:

```
$ R CMD BATCH --args <sample name>.bincount.txt <sample name> \
     <GC content>.txt <bad bins>.txt cnv/cbs.r <outfile>.txt
```

Scripts to generate CNV profiles After alignment, CNV profiles can be directly generated using the script:

```
$ ./cnv/cnv-<5k/20k/50k>.sh <aligned>.sam <sample name>
```

This script executes the three steps described above to generate the profiles. The script requires the environment variable `SMURFDIR` set to the location of the smurfseq-scripts repository, using the command:

```
$ export SMURFDIR=<path to smurfseq-scripts dir>
```

4 Miscellaneous analysis of sequenced reads and mapped fragments

4.1 Read length distribution

After the sequencing experiment using SMURF-seq, the sequenced read length distribution can be used to determine the quality of the library. A "good" library would have majority of reads in the kilobase range. The presence of a large quantity of reads in the $\sim$600 bp range would indicate the ligation step was not efficient and there is an abundance of molecules that were not ligated.

```
$
```

4.2 Fragment length distribution

After aligning the reads, the length of the mapped fragments can be obtained using:

```
$
```

4.3 Closeness to RE sites

SMURF-seq protocol used a restriction enzyme to fragment DNA molecules prior to ligation. Thus after aligning SMURF-seq reads, the fragments would align to restriction sites on the reference genome. However, due to single nucleotide or structural variations in the sample sequenced, there could be fragments that do not align to a restriction site on the reference genome.

The locations of the restriction sites on the reference genome can be generated using the command:

```
$
```

The distribution of the distance of the mapped fragments from its closest restriction site on the reference genome can be generated using:

```
$
```

4.4 Extracting unaligned bases from reads

After aligning the reads, there would be substring of reads that were not aligned. These unaligned substring can be extracted using the command:

```
$
```

References

[1] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[2] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 1:7, 2018.

[3] Szymon M Kielbasa, Raymond Wan, Kengo Sato, Paul Horton, and Martin Frith. Adaptive seeds tame genomic sequence comparison. *Genome research*, pages gr–113985, 2011.

[4] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[5] Timour Baslan, Jude Kendall, Linda Rodgers, Hilary Cox, Mike Riggs, Asya Stepansky, Jennifer Troge, Kandasamy Ravi, Diane Esposito, B Lakshmi, et al. Genome-wide copy number analysis of single cells. *Nature protocols*, 7(6):1024, 2012.

[6] Jude Kendall and Alexander Krasnitz. *Computational Methods for DNA Copy-Number Analysis of Tumors*, pages 243–259. Springer New York, New York, NY, 2014.

[7] Erik Gerdtsson, Milind Pore, Jana-Aletta Thiele, Anna Sandström Gerdtsson, Paymaneh D Malihi, Rafael Nevarez, Anand Kolatkar, Carmen Ruiz Velasco, Sophia Wix, Mohan Singh, et al. Multiplex protein detection on circulating tumor cells from liquid biopsies using imaging mass cytometry. *Convergent Science Physical Oncology*, 4(1):015002, 2018.

[8] Paymaneh D Malihi, Michael Morikado, Lisa Welter, Sandy T Liu, Eric T Miller, Radu M Cadaneanu, Beatrice S Knudsen, Michael S Lewis, Anders Carlsson, Carmen Ruiz Velasco, et al. Clonal diversity revealed by morphoproteomic and copy number profiles of single prostate cancer cells at diagnosis. *Convergent Science Physical Oncology*, 4(1):015003, 2018.

[9] Adam B Olshen, ES Venkatraman, Robert Lucito, and Michael Wigler. Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics*, 5(4):557–572, 2004.

[10] Venkatraman E Seshan, Adam B Olshen, et al. Dnacopy: a package for analyzing dna copy data. 2010.