

Note S1 Methods

Note S2 Experiment setup

S2.1 Datasets

Note S3 Results

Smash++ and several other methods have been carried out on a collection of synthetic and real sequences. The machine used for the tests had an 8-core 3.40 GHz Intel[®] Core[™] i7-6700 CPU with 32 GB RAM.

Note S4 Tool availability and implementation

Smash++ is implemented in C++ language and is available at [1]. This tool is able to find rearrangements in sequences, Fasta and Fastq files; although, it is highly recommended to use sequences as input. To work with Smash++, it should be first installed, and then run. In the following sections, we describe installing and running Smash++.

S4.1 Install

In order to install Smash++, we run the following commands:

```
1 git clone https://github.com/smortezah/smashpp.git
2 cd smashpp
3 cmake .
4 make
```

S4.2 Run

Running

```
1 ./smashpp
```

provides the following guide:

```
1 SYNOPSIS
2   ./smashpp [OPTIONS]...  -r [REF-FILE] -t [TAR-FILE]
3
4 SAMPLE
5
6 DESCRIPTION
7   Mandatory arguments:
8     -r, --ref FILE           reference file (Seq/Fasta/Fastq)
9     -t, --tar FILE           target file      (Seq/Fasta/Fastq)
10
11   Options:
12     -v, --verbose            more information
13     -l, --level INT          level of compression [0;4]      COMPRESS
14     -e, --ent-n FLOAT        Entropy of 'N's [0.0;100.0]      COMPRESS
15     -n, --nthr INT           number of threads [1;8]
16     -fs, --filter-scale S|M|L scale of the filter {S|small,    FILTER
17                               M|medium, L|large}
18     -w, --wsize INT          window size [1;100000]          FILTER
19     -wt, --wtype [0;7]       type of windowing function      FILTER
20                               {0|rectangular, 1|hamming, 2|hann,
21                               3|blackman, 4|triangular, 5|welch,
22                               6|sine, 7|nuttall}
23     -d, --step INT           sampling steps                  FILTER
24     -th, --thresh FLOAT      threshold [0.0;20.0]           FILTER
25     -sp, --save-profile       save profile (*.prf)           SAVE
26     -sf, --save-filter        save filtered file (*.fil)     SAVE
27     -sb, --save-seq           save sequence (input: Fasta/Fastq) SAVE
28     -ss, --save-segment       save segmented files (*-s )    SAVE
29     -sa, --save-all          save profile, filetered and    SAVE
30                               segmented files
31     -h, --help               usage guide
32     -rm, --ref-model [ , [ , , ],ir,α, / ,ir, , :... ]
33 MODEL
34     -tm, --tar-model [ , [ , , ],ir, , / ,ir, , :... ]
35 MODEL
36
37   parameters of models
38   (INT) : context size
39   (INT) : width of sketch in log2 form,
40           e.g., set 10 for w=2^10=1024
41   (INT) : depth of sketch
42   ir: inverted repeat {0, 1, 2}
```

```
40                                     0: regular (not inverted)
41                                     1: inverted, solely
42                                     2: both regular and inverted
43             (FLOAT)                : estimator
44             (FLOAT)                : forgetting factor [0.0;1.0)
45             (INT)                   : threshold (no. substitutions)
```

S4.3 Example

This section guides, step-by-step, employing Smash++ to find rearrangements.

Install Smash++ and provide the required files

First, we install Smash++:

```
1 git clone https://github.com/smortezah/smashpp.git
2 cd smashpp
3 cmake .
4 make
```

Then, we copy Smash++'s binary file into `example/` directory and go to that directory:

```
1 cp smashpp example/
2 cd example/
```

References

- [1] M. Hosseini, D. Pratas, and A. J. Pinho. Smash++. [Online]. Available: <https://github.com/smortezah/smashpp>