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. In the following sections, we describe installing and running Smash++.

S4.1 Install

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

```
git clone https://github.com/smortezah/smashpp.git
cd smashpp
cmake .
make
```

S4.2 Run

Running

```
./smashpp
```

provides the following guide:

```
SYNOPSIS
  ./smashpp [OPTIONS]...  -r [REF-FILE] -t [TAR-FILE]

SAMPLE

DESCRIPTION
  Mandatory arguments:
  -r,  --ref FILE            reference file (Seq/Fasta/Fastq)
  -t,  --tar FILE            target file    (Seq/Fasta/Fastq)

  Options:
  -v,  --verbose             more information
  -l,  --level INT           level of compression [0;4]     COMPRESS
  -e,  --ent-n FLOAT         Entropy of 'N's [0.0;100.0]    COMPRESS
  -n,  --nthr  INT           number of threads [1;8]
  -fs, --filter-scale S|M|L  scale of the filter {S|small,    FILTER
                             M|medium, L|large}
  -w,  --wsize INT           window size [1;100000]           FILTER
  -wt, --wtype [0;7]         type of windowing function       FILTER
                             {0|rectangular, 1|hamming, 2|hann,
                             3|blackman, 4|triangular, 5|welch,
                             6|sine, 7|nuttall}
  -d,  --step   INT          sampling steps                   FILTER
  -th, --thresh FLOAT        threshold [0.0;20.0]             FILTER
  -sp, --save-profile        save profile (*.prf)               SAVE
  -sf, --save-filter         save filtered file (*.fil)         SAVE
  -sb, --save-seq            save sequence (input: Fasta/Fastq) SAVE
  -ss, --save-segment        save segmented files (*- s   )       SAVE
  -sa, --save-all            save profile, filetered and      SAVE
                             segmented files
  -h,  --help                usage guide
  -rm, --ref-model [    ,[    ,    ,]ir,α,    /    ,ir,    ,    :...]
MODEL
  -tm, --tar-model [    ,[    ,    ,]ir,    ,    /    ,ir,    ,    :...]
MODEL
                             parameters of models
                       (INT)     :  context size
                       (INT)     :  width of sketch in log2 form,
                                 e.g., set 10 for w=2^10=1024
                       (INT)     :  depth of sketch
                             ir: inverted repeat {0, 1, 2}
```

```
                                    0: regular (not inverted)
                                    1: inverted, solely
                                    2: both regular and inverted
                        (FLOAT)     :  estimator
                        (FLOAT)     :  forgetting factor [0.0;1.0)
                          (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++:

```
git clone https://github.com/smortezah/smashpp.git
cd smashpp
cmake .
make
```

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

```
cp smashpp example/
cd example/
```

References

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