Note S1 Tool availability and implementation

Smash++ is implemented in `C++` language and is available at [1]. This tool is able to find and visualize rearrangements in sequences, FASTA and FASTQ files; although, it is highly recommended to use sequences as input. In the following sections, we describe installing and running Smash++.

S1.1 Install

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

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

S1.2 Run

A reference file and a target file are clearly mandatory to run Smash++ (without visualization). 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, 5]
  -m,  --min   INT            min segment size: [1, 4294967295]
  -nr, --no-redun             do NOT compute self complexity
  -e,  --ent-n FLOAT          Entropy of 'N's: [0.0, 100.0]
  -n,  --nthr  INT            number of threads: [1, 8]
  -fs, --filter-scale S|M|L   scale of the filter:
                              {S|small, M|medium, L|large}
  -w,  --wsize INT            window size: [1, 4294967295]
  -wt, --wtype INT/STRING     type of windowing function:
                              {0|rectangular, 1|hamming, 2|hann,
                              3|blackman, 4|triangular, 5|welch,
                              6|sine, 7|nuttall}
  -d,  --step   INT           sampling steps
  -th, --thresh FLOAT         threshold: [0.0, 20.0]
  -sb, --save-seq             save sequence (input: Fasta/Fastq)
  -sp, --save-profile         save profile (*.prf)
  -sf, --save-filter          save filtered file (*.fil)
  -ss, --save-segment         save segmented files (*-s_i)
  -sa, --save-all             save profile, filetered and
                              segmented files
  -h,  --help                 usage guide
  -rm, --ref-model  k,[w,d,]ir,a,g/t,ir,a,g:...
  -tm, --tar-model  k,[w,d,]ir,a,g/t,ir,a,g:...
                              parameters of models
                        (INT) k:  context size
                        (INT) w:  width of sketch in log2 form,
                                  e.g., set 10 for w=2^10=1024
                        (INT) d:  depth of sketch
                        (INT) ir: inverted repeat: {0, 1, 2}
```

```
                                  0: regular (not inverted)
                                  1: inverted, solely
                                  2: both regular and inverted
                       (FLOAT) a:  estimator
                       (FLOAT) g:  forgetting factor: [0.0, 1.0)
                         (INT) t:  threshold (no. substitutions)
```

The arguments “-r” and “-t” are used to specify the reference and the target, respectively, which are highly recommended to have short names. Level of compression, that is an integer between 0 and 5, can be determined with “-l”. By setting “-m” to an integer value, only those regions in the reference file that are greater than that value can be considered for the compression. Triggering “-nr” makes the tool not to perform the reference-free compression (self-complexity computation) part.

In implementation of the reference-based compression, we have replaced ‘N’ bases in the references and the targets with ‘A’s and ‘T’s, respectively. On reference-free compression, they are replaced with ‘A’s, in both references and targets. If a user tends to replace ‘N’ bases in a sequence with a normal distribution of ‘A’, ‘C’, ‘G’ and ‘T’s, he/she can employ GOOSE toolkit [2]. Note that we have set the entropy of ‘N’s to 2.0, by default, but it is possible for the user to set them to another value of interest, by “-e” option.

Building different finite-context models can be done in the multi-threaded fashion, setting “-n” to an integer. To find similar regions in the reference and the target, information profile (obtained by compression) needs to be filtered, of which the scale can be set as S (small), M (medium) or L (large). Size of the window and type of the windowing function, described in **??**, can be set by “-w” and “-wt” options, respectively. Instead of considering the complete profile information, the user is able to make samples of it by steps of which size can be determined by “-d”. For the purpose of segmenting the filtered information profile, the average entropy of reference-based compression is used as the threshold, by default. However, this threshold can be altered by “-th” option.

Smash++ accepts FASTA and FASTQ files as input, in addition to sequences. In these cases, the input files are converted to sequences and then processed further. It is possible to save these sequences by “-sb” option.

After obtaining the information profile, Smash++ filters it and then removes it, by default. However, it is possible to save the profile by “-sp” option. The same thing happens to the filtered file, i.e., it is segmented and then is removed. But, the user can use “-sf” to save the filtered file. Also, the segmented files can be saved using “-ss”. The user can save all the information profile, filtered and segmented files, by triggering “-sa” option.

For the purpose of compression, either reference-based or reference-free, it is recommended to use “-l” option, since it configures the models automatically. However, using “-rm” and “-tm”, the user would be able to manually configure the reference model, for reference-based compression, and the target model, for reference-free compression. Parameters of the models are described in detail in section **??**.

Running Smash++ (without visualization), positions of the similar regions in the reference and the target, and also complexity of the regions is saved in a “*.pos” file. To visualize this file, one can type

```
./smashpp -viz
```

which gives

```
SYNOPSIS
  ./smashpp -viz  OPTIONS...  -o SVG-FILE  POS-FILE

SAMPLE

DESCRIPTION
  Mandatory arguments:
  POS-FILE                 positions file, generated by
                           Smash++ tool (*.pos)

  Options:
  -v,  --verbose           more information
  -o,  --out SVG-FILE      output image name (*.svg)
  -rn, --ref-name STRING   reference name shown on output. If name
                           has space, use "s, e.g. "Seq label".
                           Default: name in header of position file.
  -tn, --tar-name STRING   target name shown on output
  -vv, --vertical          vertical view
  -nn, --no-nrc            do NOT show normalized
                           relative compression (NRC)
  -nr, --no-redun          do NOT show self complexity
  -ni, --no-inv            do NOT show inverse maps
  -ng, --no-reg            do NOT show regular maps
  -l,  --link     INT      type of the link between maps: [1, 6]
  -c,  --color    INT      color mode: [0, 2]
  -p,  --opacity  FLOAT    opacity: [0.0, 1.0]
  -w,  --width    INT      width of the sequence: [15, 100]
  -s,  --space    INT      space between sequences: [15, 200]
  -f,  --mult     INT      multiplication factor for
                           color ID: [1, 255]
  -b,  --begin    INT      beginning of color ID: [0, 255]
  -rt, --ref-tick INT      reference tick: [1, 4294967295]
  -tt, --tar-tick INT      target tick: [1, 4294967295]
  -th, --tick-human 0|1    tick human readable: 0=false, 1=true
  -m,  --min      INT      minimum block size: [1, 4294967295]
  -h,  --help              usage guide
```

S1.3 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

[2] D. Pratas. Goose. [Online]. Available: https://github.com/pratas/goose