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 the Smash++ tool.

S1.1   Install

To install Smash++ on various operating systems, follow the instructions below. Note that the precompiled executables are available for 64 bit operating systems in the "bin/" directory.

Linux

- Install "git" and "cmake":

```
1  sudo apt update
2  sudo apt install git cmake
```

- Clone Smash++ and install it:

```
1  git clone https://github.com/smortezah/smashpp.git
2  cd smashpp
3  ./install.sh
```

macOS

- Install "Homebrew", "git" and "cmake":

```
1  /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
       install/master/install)"
2  brew install git cmake
```

- Clone Smash++ and install it:

```
1  git clone https://github.com/smortezah/smashpp.git
2  cd smashpp
3  ./install.sh
```

Windows

- Download and install "CMake", e.g., from https://github.com/Kitware/CMake/releases/download/v3.14.4/cmake-3.14.4-win64-x64.msi. Make sure to add it to the system PATH. For example, if CMake is installed in "C:\Program Files", add "C:\Program Files\CMake\bin" to the system PATH.

- Download and install "mingw-w64", e.g., from https://sourceforge.net/projects/mingw-w64/files/latest/download. Make sure to add it to the system PATH. For example, if it is installed in "C:\mingw-w64", add "C:\mingw-w64\mingw64\bin" to the system PATH.

- Download and install "git", from https://git-scm.com/download/win.

- Clone Smash++ and install it:

```
1  git clone https://github.com/smortezah/smashpp.git
2  cd smashpp
3  .\install.bat
```

S1.2   Run

A reference file and a target file are clearly mandatory to run Smash++ (without visualization). Running

```
1   ./smashpp
```

provides the following guide:

```
1   SYNOPSIS
2     ./smashpp [OPTIONS]  -r <REF-FILE>  -t <TAR-FILE>
3
4   OPTIONS
5     Required:
6     -r  <FILE>         = reference file (Seq/FASTA/FASTQ)
7     -t  <FILE>         = target file    (Seq/FASTA/FASTQ)
8
9     Optional:
10    -l  <INT>          = level of compression: [0, 5]. Default -> 0
11    -m  <INT>          = min segment size: [1, 4294967295]      -> 50
12    -e  <FLOAT>        = entropy of 'N's: [0.0, 100.0]          -> 2.0
13    -n  <INT>          = number of threads: [1, 8]              -> 4
14    -f  <INT>          = filter size: [1, 4294967295]           -> 256
15    -ft <INT/STRING>   = filter type (windowing function):      -> hann
16                         {0|rectangular, 1|hamming, 2|hann,
17                          3|blackman, 4|triangular, 5|welch,
18                          6|sine, 7|nuttall}
19    -fs [S][M][L]      = filter scale:                          -> L
20                         {S|small, M|medium, L|large}
21    -d  <INT>          = sampling steps                         -> 1
22    -th <FLOAT>        = threshold: [0.0, 20.0]                 -> 1.5
23    -rb <INT>          = ref beginning guard: [-32768, 32767]   -> 0
24    -re <INT>          = ref ending guard: [-32768, 32767]      -> 0
25    -tb <INT>          = tar beginning guard: [-32768, 32767]   -> 0
26    -te <INT>          = tar ending guard: [-32768, 32767]      -> 0
27    -dp                = deep compression                       -> no
28    -nr                = do NOT compute self complexity         -> no
29    -sb                = save sequence (input: FASTA/FASTQ)     -> no
30    -sp                = save profile (*.prf)                   -> no
31    -sf                = save filtered file (*.fil)             -> no
32    -ss                = save segmented files (*.s[i])          -> no
33    -sa                = save profile, filetered and            -> no
34                         segmented files
35    -rm k,[w,d,]ir,a,g/t,ir,a,g:...
36    -tm k,[w,d,]ir,a,g/t,ir,a,g:...
37                       = parameters of models
38              <INT>  k:  context size
39              <INT>  w:  width of sketch in log2 form,
40                         e.g., set 10 for w=2^10=1024
41              <INT>  d:  depth of sketch
42              <INT>  ir: inverted repeat: {0, 1, 2}
43                         0: regular (not inverted)
44                         1: inverted, solely
45                         2: both regular and inverted
46            <FLOAT>  a:  estimator
47            <FLOAT>  g:  forgetting factor: [0.0, 1.0)
48              <INT>  t:  threshold (no. substitutions)
49    -ll                = list of compression levels
50    -h                 = usage guide
51    -v                 = more information
52    --version          = show version
```

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 bigger than that value can be considered for 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 can be set by "-w" and "-wt" options, respectively. Besides Hann window that is used as default to filter the profile information, we have implemented several other window functions, including Blackman [3], Hamming [4], Nuttall [5], rectangular [6], sine [7], triangular [8] and Welch [9] windows. These functions are given by

$$w[n] = 1, \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\text{(rectangular)}$$

$$w[n] = 1 - \left| \frac{n - N/2}{L/2} \right|, \quad L = N, \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\text{(triangular/Bartlett)}$$

$$w[n] = 1 - \left(\frac{n - N/2}{N/2} \right)^2, \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\text{(Welch)}$$

$$w[n] = \sin\left(\frac{\pi n}{N} \right), \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\text{(sine)}$$

$$w[n] = 0.54348 - 0.45652 \, \cos\left(\frac{2\pi n}{N} \right), \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\text{(Hamming)}$$

$$w[n] = 0.42659 - 0.49656 \, \cos\left(\frac{2\pi n}{N} \right) + 0.07685 \, \cos\left(\frac{4\pi n}{N} \right), \quad\quad\text{(Blackman)}$$

$$w[n] = 0.35577 - 0.48740 \, \cos\left(\frac{2\pi n}{N} \right) + 0.14423 \, \cos\left(\frac{4\pi n}{N} \right) - 0.01260 \, \cos\left(\frac{6\pi n}{N} \right), \quad\text{(Nuttall)}$$

(Eq. S1)

and are plotted in Fig. S1. 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.
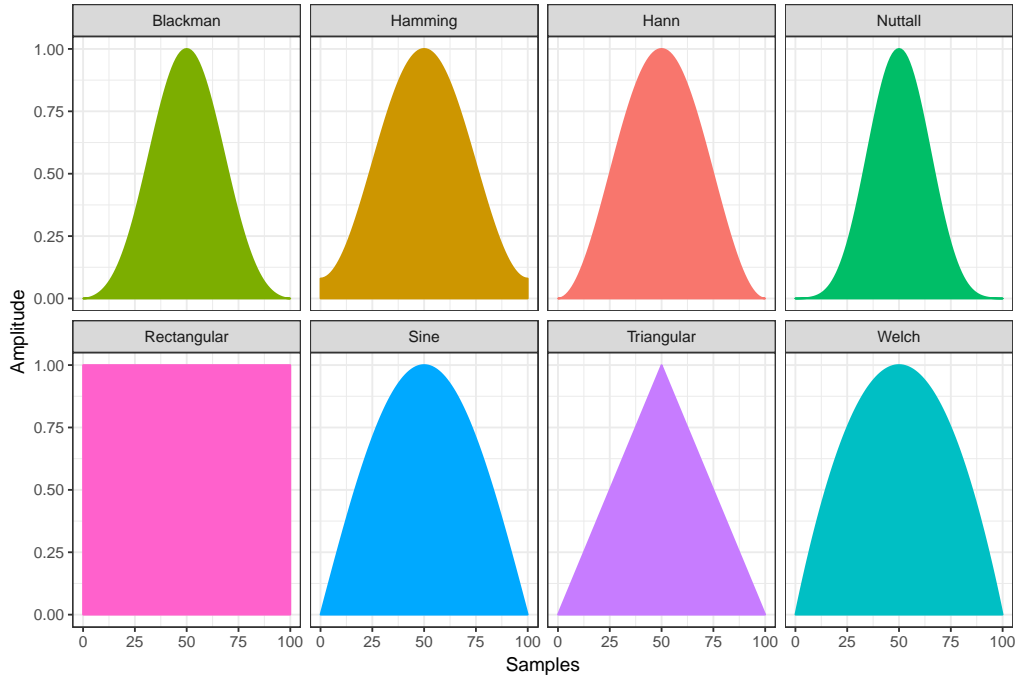
Fig. S1. Window functions.

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 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

```
1    ./smashpp -viz
```

which gives

```
1    SYNOPSIS
2    ./smashpp -viz  OPTIONS...  -o SVG-FILE  POS-FILE
3
4    SAMPLE
5
6    DESCRIPTION
7    Mandatory arguments:
8    POS-FILE                positions file, generated by
9    Smash++ tool (*.pos)
10
11   Options:
12   -v,  --verbose          more information
13   -o,  --out SVG-FILE     output image name (*.svg)
14   -rn, --ref-name STRING  reference name shown on output. If name
15   has space, use "s, e.g. "Seq label".
16   Default: name in header of position file.
17   -tn, --tar-name STRING  target name shown on output
18   -vv, --vertical         vertical view
19   -nn, --no-nrc           do NOT show normalized
20   relative compression (NRC)
21   -nr, --no-redun         do NOT show self complexity
22   -ni, --no-inv           do NOT show inverse maps
23   -ng, --no-reg           do NOT show regular maps
24   -l,  --link     INT     type of the link between maps: [1, 6]
25   -c,  --color    INT     color mode: [0, 2]
26   -p,  --opacity  FLOAT   opacity: [0.0, 1.0]
27   -w,  --width    INT     width of the sequence: [15, 100]
28   -s,  --space    INT     space between sequences: [15, 200]
29   -f,  --mult     INT     multiplication factor for
30   color ID: [1, 255]
31   -b,  --begin    INT     beginning of color ID: [0, 255]
32   -rt, --ref-tick INT     reference tick: [1, 4294967295]
33   -tt, --tar-tick INT     target tick: [1, 4294967295]
34   -th, --tick-human 0|1   tick human readable: 0=false, 1=true
35   -m,  --min      INT     minimum block size: [1, 4294967295]
36   -h,  --help             usage guide
```

The output of Smash++ visualizer is an "SVG" file, which its name is determined by "-o" option. By default, it is named "map.svg". Names of the reference and the target, which are going to be printed on the output image, can be altered by "-rn" and "-tn", respectively. They are by default the names written in the positions file. To have a vertical view of the image, instead of the default horizontal view, one can use "-vv" trigger.

Smash++ performs reference-based and reference-free compressions to calculate the normalized relative compression (NRC) and redundancy (self complexity), respectively. If the user is not interested in showing them, he/she can turn them off by "-nn" and "-nr" triggers. In addition, Smash++ considers both regular and reverse complement maps by default in its calculations. Triggering "-ni" and "-ng" will stop showing inverted and regular maps, respectively.

Options "-l", "-c", "-p", "-w", "-s", "-f" and "-b" can be used to change the appearance of the image. Assigning integers to "-rt" and "-tt" options will change the tick sizes of the reference and the target, respectively. Smash++ prints the sizes on axes in human readable format, e.g., 1K, 2M, etc. However, it can be triggered by "-th" option. Note that, here, 1K is equivalent to 1000 and not 1024. Finally, by setting "-m" to an integer value, only the regions that are bigger than that value will be illustrated.

S1.3 Example

This section guides, step-by-step, employing Smash++ to find and visualize rearrangements in a sample genomic data.

Install Smash++ and provide the required files

First, we install Smash++:

```
1   git clone https://github.com/smortezah/smashpp.git
2   cd smashpp
3   ./install.sh
```
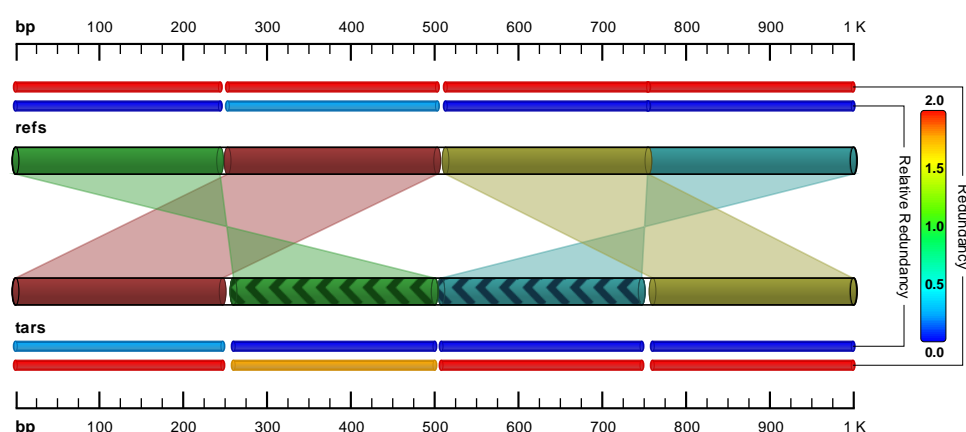
Then, we copy `smashpp` binary file into `example/` directory and go to that directory:

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

In this directory, a 1000 byte reference sequence, `ref`, and a 1000 byte target sequence, `tar`, are provided. Running

```
1   ./smashpp -r refs -t tars -w 45 -l 3
2   ./smashpp -viz refs.tars.pos
```

results in Fig. S2, which is saved as "map.svg".

Fig. S2. The result of running Smash++ on ...

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

[3] R. Blackman and J. Tukey, "Particular pairs of windows," *The measurement of power spectra, from the point of view of communications engineering*, pp. 95–101, 1959.

[4] J. W. Tukey and R. W. Hamming, *Measuring noise color*. Bell Telephone Laboratories, 1949.

[5] A. Nuttall, "Some windows with very good sidelobe behavior," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 1, pp. 84–91, 1981.

[6] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 1999.

[7] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[8] M. S. Bartlett, "Periodogram analysis and continuous spectra," *Biometrika*, vol. 37, no. 1/2, pp. 1–16, 1950.

[9] P. Welch, "The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.