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":

```
sudo apt update
sudo apt install git cmake
```

- Clone Smash++ and install it:

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

macOS

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

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

- Clone Smash++ and install it:

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

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

S1.2 Run Smash++

Various options are provided with the interface of the proposed tool, which are described in Table **??**.

$$
\begin{aligned}
&w[n] = 1, && \text{(rectangular)}\\
&w[n] = 1 - \left|\tfrac{n-N/2}{L/2}\right|, \quad L = N, && \text{(triangular/Bartlett)}\\
&w[n] = 1 - \left(\tfrac{n-N/2}{N/2}\right)^2, && \text{(Welch)}\\
&w[n] = \sin\left(\tfrac{\pi n}{N}\right), && \text{(sine)}\\
&w[n] = 0.54348 - 0.45652\ \cos\left(\tfrac{2\pi n}{N}\right), && \text{(Hamming)}\\
&w[n] = 0.42659 - 0.49656\ \cos\left(\tfrac{2\pi n}{N}\right) + 0.07685\ \cos\left(\tfrac{4\pi n}{N}\right), && \text{(Blackman)}\\
&w[n] = 0.35577 - 0.48740\ \cos\left(\tfrac{2\pi n}{N}\right) + 0.14423\ \cos\left(\tfrac{4\pi n}{N}\right) - 0.01260\ \cos\left(\tfrac{6\pi n}{N}\right), && \text{(Nuttall)}\\
& && \textbf{(Eq. S1)}
\end{aligned}
$$

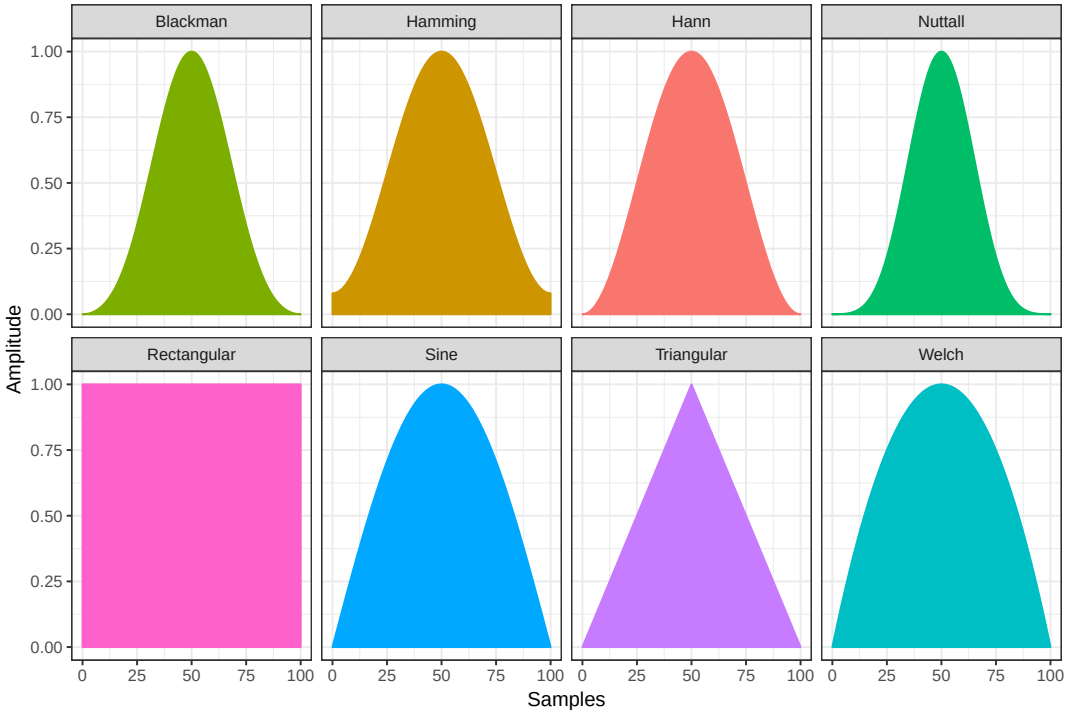

Fig. S1. Various windowing functions implemented and embedded in Smash++.

By running Smash++, positions of the similar regions in reference and target sequences, and also complexity of the regions is saved in a ".pos" file. This tab-separated file has a header including:

1. The string "#SMASHPP" as a specifier for the Smash++ tool,
2. The name of reference sequence,
3. The size of reference sequence,
4. The name of target sequence,
5. The size of target sequence,

and a body including:

1. Initial position of the reference sequence,
2. Final position of the reference sequence,
3. Average entropy of compressing the associated target block considering this reference block as the reference,

4. Complexity (average entropy) of the detected reference block, calculated by reference-free compression,
5. Initial position of the target sequence,
6. Final position of the target sequence,
7. Average entropy of compressing the associated reference block considering this target block as the reference,
8. Complexity of the detected target block, calculated by reference-free compression.

As an example, the header of the following ".pos" file shows that the reference "Ref" and the target "Tar" are 5,000,000 bases long. The body shows that there is a block in the Ref, from the position 1250013 up to 2499988, which is similar to a block in the Tar, from the position 1250017 to 2499970. Average entropy of compressing the Tar block, using the Ref block as reference, is 0.238067. This number is 0.26698 when the Ref block is compressed based on the model of the Tar block. Also, complexities of the Ref and the Tar blocks are 1.97158 and 1.98884, respectively.

```
#SMASHPP Ref       5000000  Tar       5000000
1250013  2499988  0.26698  1.97158  1250017  2499970  0.238067 1.98884
```

S1.3 Run Smash++ visualizer

Running Smash++ (without visualization), positions of the similar regions in reference and target sequences, and also complexity of the regions is saved in a *.pos file, that can be visualized by

```
./smashpp -viz
```

which gives

```
SYNOPSIS
  ./smashpp -viz [OPTIONS]  -o <SVG-FILE>  <POS-FILE>

OPTIONS
  Required:
  <POS-FILE>          = position file, generated by
                        Smash++ tool (*.pos)

  Optional:
  -o  <SVG-FILE>      = output image name (*.svg)             -> map.svg
  -rn <STRING>        = reference name shown on output. If it
                        has space, use double quotes, e.g.
                        "Seq label". Default: name in header
                        of position file
  -tn <STRING>        = target name shown on output
  -l  <INT>           = type of the link between maps: [1, 6] -> 1
  -c  <INT>           = color mode: [0, 1]                    -> 0
  -p  <FLOAT>         = opacity: [0.0, 1.0]                   -> 0.9
  -w  <INT>           = width of the sequence: [15, 100]      -> 16
  -s  <INT>           = space between sequences: [15, 200]    -> 62
  -f  <INT>           = multiplication factor for             -> 43
                        color ID: [1, 255]
  -b  <INT>           = beginning of color ID: [0, 255]       -> 0
  -rt <INT>           = reference tick: [1, 4294967295]
  -tt <INT>           = target tick: [1, 4294967295]
  -th [0][1]          = tick human readable: 0=false, 1=true  -> 1
  -m  <INT>           = minimum block size: [1, 4294967295]   -> 1
  -vv                 = vertical view                         -> no
  -nn                 = do NOT show normalized relative       -> no
                        compression (NRC)
  -nr                 = do NOT show self complexity           -> no
  -ni                 = do NOT show inverse maps              -> no
  -ng                 = do NOT show regular maps              -> no
  -h                  = usage guide
  -v                  = more information
  --version           = show version
```

The output of Smash++ visualizer is an "SVG" file whose 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.

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, and so on.

By setting “-m” to an integer value, only the regions that are bigger than that value will be illustrated. 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 by default both regular and reverse complement maps in its calculations. Triggering “-ni” and “-ng” will stop showing inverted and regular maps, respectively.

S1.4 Example

This section guides step-by-step employing Smash++ to find and visualize rearrangements in a sample genomic data. Note that the commands can be run on Linux and macOS, however, they are similar in Windows.

Install Smash++ and provide the required files

First, install Smash++:

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

Then, copy `smashpp` executable file into `example/` directory and go to that directory:

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

There is a 1000 byte reference sequence, named `refs`, and a 1000 byte target sequence, named `tars`, in this directory. Running

```
./smashpp -r refs -t tars -f 45 -l 3
./smashpp -viz -p 1 -s 50 -w 15 refs.tars.pos
```

results in Fig. S2, which has been saved as “map.svg”.

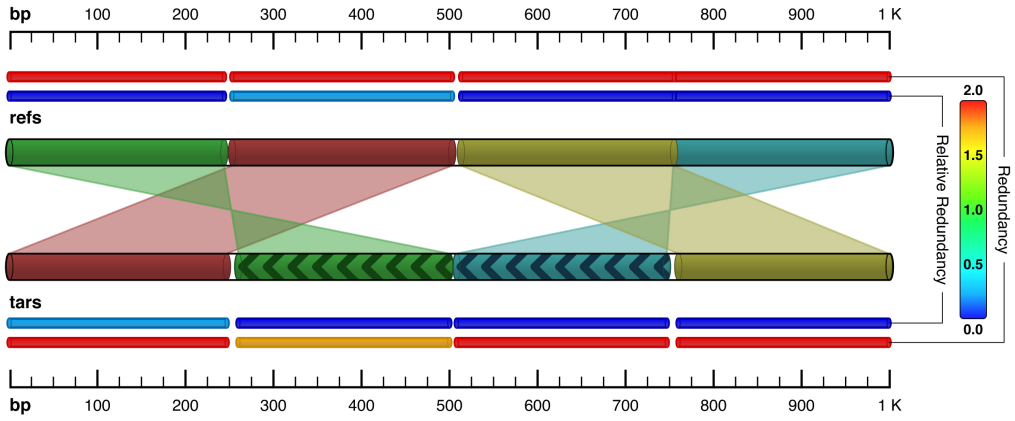

Fig. S2. An example of running Smash++ on two 1000 base sequences. Two similar regions in regular mode and two similar ones in inverted mode are detected.