

## Note S1 Tool availability and implementation

Smash++ is implemented in C++ language and is publicly available at [1], under GNU GPL v3 license. This tool is able to find and visualize rearrangements in a pair of DNA sequences. It is recommended to use as input bare sequences, i.e., without header or quality scores, although, FASTA and FASTQ formats are also supported. 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.

#### Conda

```
1 conda install -c cobilab smashpp
```

#### 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 Smash++

Various options are provided with the interface of the proposed tool, which are described in Table S1. The commands for running Smash++ have the form of the following:

```
1 ./smashpp [OPTIONS] -r <REF-FILE> -t <TAR-FILE>
```

**Table S1.** Options provided by Smash++ interface.

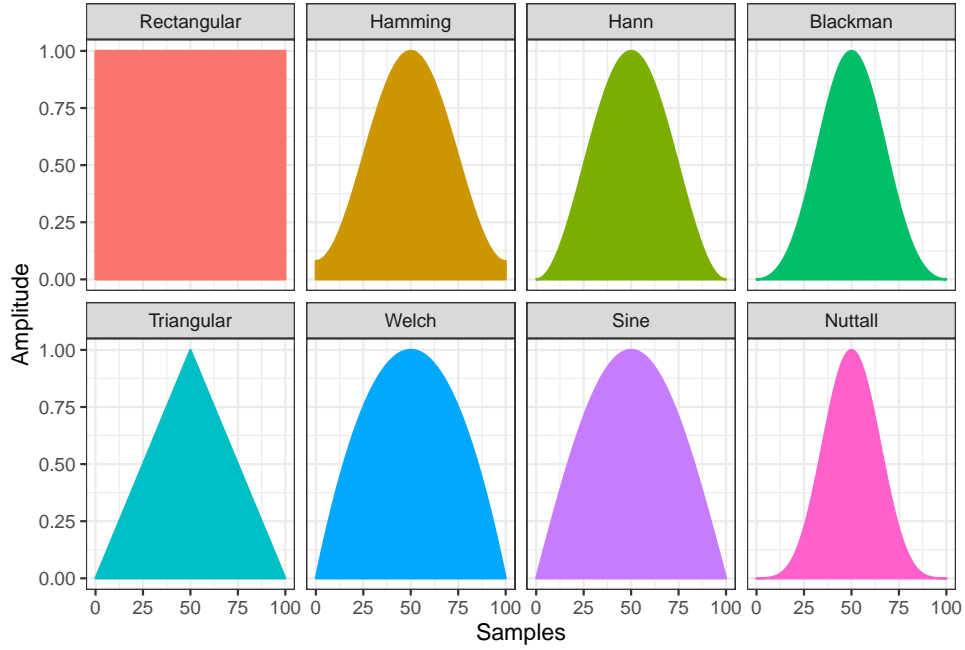
Flag	Input	Description
<i>Required</i>		
-r	Seq/FASTA/FASTQ	Reference file. Spaces in the name is supported.
-t	Seq/FASTA/FASTQ	Target file. Spaces in the name is supported. It is recommended to have short names.
<i>Optional</i>		
-l	Integer: [0, 6] Default: 3	Level of compression.
-m	Integer: $[1, 2^{32} - 1]$ Default: 50	Minimum segment size. Only the regions that have greater sizes than this value would be considered for compression.
-e	Float: [0.0, 100.0] Default: 2.0	Entropy of 'N' bases. In implementation of the reference-based compression, we replace 'N's in references and targets with 'A's and 'T's, respectively. On reference-free compression, we replace them 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].
-n	Integer: [1, 255] Default: 4	Number of threads. Creating multiple finite-context models and substitution-tolerant Markov models can be done in a multi-threaded fashion.
-f	Integer: $[1, 2^{32} - 1]$ Default: 256	Filter size. In the process of finding similar regions in the reference and the target sequences, the information content that would be obtained by compression needs to be filtered.
-ft	Integer or String: {0/rectangular, 1/hamming, 2/hann, 3/blackman, 4/triangular, 5/welch, 6/sine, 7/nuttall} Default: hann	Filter type (windowing function). Besides Hann window that is used by default to smooth the information content (profile), we have implemented several other windowing functions: Blackman [3], Hamming [4], Nuttall [5], rectangular [6], sine [7], triangular [8] and Welch [9] windows. These functions along with Hann function are given by Equation S1 and are plotted in Fig. S1.
-fs	Char or String: {S/small, M/medium, L/large}	Filter scale. It automatically chooses filter size. If a user does not use this flag, the "-f" flag will handle the filtering size.
-d	Integer: $[1, 2^{64} - 1]$ Default: 1	Sampling steps. Instead of considering the whole information content, this option can be used to make samples of it.
-th	Float: [0.0, 20.0] Default: 1.5	Threshold for entropy. This option can be used for segmenting the filtered information content.
-rb	Integer: $[-2^{15}, 2^{15} - 1]$	Reference beginning guard.
-re	Integer: $[-2^{15}, 2^{15} - 1]$	Reference ending guard.
-tb	Integer: $[-2^{15}, 2^{15} - 1]$	Target beginning guard.
-te	Integer: $[-2^{15}, 2^{15} - 1]$	Target ending guard.

Default: 0		Smash++ is capable of finding even very small similar regions in two sequences. However, we have found experimentally that when it is running in a very sensitive mode, there might be some cases in which the size of similar regions in the reference and the target are not balanced. These cases can be handled by “-rb”, “-re”, “-tb” and “-te” options, that can resize the beginning and ending guards of reference and target regions, respectively. For example, if “-tb 10” is used, the first 10 bases in each target region will be discarded, which results in smaller regions. Note that when the guard sizes of target regions are increased, the models built from these regions would be slightly different than the original models; consequently, the sizes of reference regions that are detected as being similar to the ones from the target would be modified, as well. Therefore, changing the guard sizes of target regions will affect the sizes of reference regions. In the case of activating deep compression, by “-dp”, changing the guard sizes of reference regions would affect the sizes of target regions, as well.
-ar	—	Consider asymmetric regions. It makes Smash++ not to balance the similar reference and target regions.
-dp	—	Deep compression. It means that similar regions in target and reference sequences are found in three phases, instead of two: 1) the model of the reference is built and the target is compressed based upon that model, 2) the model of each detected region is built and the whole reference is compressed based on these models and 3) the model of each detected reference region is built and the corresponding target regions will be compressed based on that model.
-nr	—	Do not compute self complexity. It makes the tool not to perform the reference-free compression (self-complexity computation).
-sb	—	Save sequence (input: FASTA/FASTQ). Smash++ accepts as input FASTA and FASTQ files, in addition to bare sequences. In these cases, the input files are first converted to sequences and then processed further. It is possible to save these sequences by this option.
-sp	—	Save profile, *.prf. When the information profile is obtained, Smash++ smoothenes then removes it, by default; however, it can be preserved by this option.
-sf	—	Save filtered file, *.fil. The filtered profile is segmented then removed, by default; however, it can be preserved by this option.
-ss	—	Save segmented files, *.s <sub>i</sub> .
-sa	—	Save all generated files, including profile, filtered and segmented files.
-rm	$k, [w, d, ]ir, \alpha, \gamma / t, ir, \alpha, \gamma : \dots$	Parameters of reference models.
-tm	$k, [w, d, ]ir, \alpha, \gamma / t, ir, \alpha, \gamma : \dots$	Parameters of target models:
	Default: 14, 0, 0.001, 0.95/5, 0, 0.001, 0.95	<ul style="list-style-type: none"> <li>• <math>k</math> (integer &gt; 1): context size,</li> <li>• <math>w</math> (integer &lt; <math>2^{64} - 1</math>): Count-Min-Log sketch width in <math>\log_2</math> form, e.g., set 10 for <math>w = 2^{10} = 1024</math>,</li> <li>• <math>d</math> (integer &gt; 0): Count-Min-Log sketch depth,</li> <li>• <math>ir</math> (integer: {0, 1, 2}): inverted repeat, including 0: regular (not inverted), 1: inverted solely, and 2: both regular and inverted,</li> <li>• <math>\alpha</math> (float &gt; 0): estimator,</li> <li>• <math>\gamma</math> (float: [0.0, 1.0]): forgetting factor,</li> <li>• <math>t</math> (integer &gt; 0): threshold (number of substitutions).</li> </ul> <p>It is recommended for compression 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, respectively. Parameters of models are described in detail in the section “Methods” of the main paper.</p>
-ll	—	List of compression levels. It shows the list of parameters that would be chosen automatically for each model.

-h	—	Usage guide.
-v	—	More information (verbose).
--version		Show the version.

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

**(Eq. S1)**



**Fig. S1.** Various windowing functions embedded in Smash++.

By running Smash++, positions of similar regions in reference and target sequences, relative redundancy and also redundancy (complexity) of the regions is saved in a “.pos” file. This tab-delimited file has a header including:

1. The string “##SMASH++” as a specifier for the Smash++ tool,
2. The “PARAM” line to list the parameters used to generate the position file,
3. The “INFO” line to provide the names and sizes of reference and target sequences,
4. The line with the names of columns of the body,

and a body including:

1. RBeg (Reference Begin): beginning of a reference region,
2. REnd (Reference End): end of a reference region,
3. RRelRdn (Reference Relative Redundancy): relative redundancy obtained by compressing the associated target block considering as reference this reference block,
4. RRdn (Reference Redundancy): redundancy (Complexity) of the detected reference block, calculated by reference-free compression,
5. TBeg (Target Begin): beginning of a target region,
6. TEnd (Target End): end of a target region,
7. TRelRdn (Target Relative Redundancy): relative redundancy obtained by compressing the associated reference block considering as reference this target block,
8. TRdn (Target Redundancy): redundancy (Complexity) of the detected target block, calculated by reference-free compression,
9. Inv (Inverted repeat): if the corresponding line is an inverted repeat. “F” (False) means it is regular and “T” (True) means it is inverted.

As an example, the header of the following “.pos” file shows that Smash++ was run by the parameters “-r dataset/REF -t dataset/TAR -l 0 -m 1000” and also the reference “REF” and the target “TAR” are 50,000 base long. The body shows that there is a reference block, from the position 5000 up to 10000, similar to a target block, from the position 2000 to 7000. Relative redundancy of compressing the TAR block, using the REF block as reference, is 1.2473. This number is 1.0187 when the REF block is compressed based on the model of the TAR block. Also, redundancies (complexities) of the REF and the TAR blocks are 1.9010 and 1.8580, respectively. The “F” in the last column shows that this similarity is not of the form inverted repeat. The second record of the body shows an inverted repeat rearrangement.

```

1  ##SMASH++
2  ##PARAM=<-r dataset/REF -t dataset/TAR -l 0 -m 1000>
3  ##INFO=<Ref=REF,RefSize=50000,Tar=TAR,TarSize=50000>
4  #RBeg  REnd  RRelRdn  RRdn  TBeg  TEnd  TRelRdn  TRdn  Inv
5  5000   10000  1.0187   1.9010  2000  7000   1.2473   1.8580  F
6  20000  30000   1.2367   1.9777  40000 30000   1.2545   1.9888  T

```

### S1.3 Run Smash++ visualizer

The position file obtained by Smash++ can be visualized by

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

The visualizer provides various options that are described in Table S2. The commands for running Smash++ visualizer are of the form

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

**Table S2.** Options provided by Smash++ visualizer interface.

Flag	Input	Description
<i>Required</i>		
	*.pos file	Position file, generated by Smash++ tool. Spaces in the name is supported. It can be redirected to Smash++ by standard input (“stdin”).
<i>Optional</i>		
-o	*.svg file Default: map.svg	Output image name.
-rn	String	Reference name shown on output.

<b>-tn</b>	String Default: names in position file's header	Target name shown on output. If it has some spaces, use double quotes, e.g. "Seq label".
<b>-l</b>	Integer: [1, 6] Default: 1	Type of the link between similar regions.
<b>-c</b>	Integer: [0, 1] Default: 0	Color mode.
<b>-p</b>	Float: [0.0, 1.0] Default: 0.9	Opacity.
<b>-w</b>	Integer: [8, 100] Default: 10	Width of the sequence.
<b>-s</b>	Integer: [5, 200] Default: 40	Space between sequences.
<b>-tc</b>	Integer: [1, 255]	Total number of colors in the map, which is automatically chosen by default.
<b>-rt</b>	Integer: [1, $2^{32} - 1$ ]	Reference tick size.
<b>-tt</b>	Integer: [1, $2^{32} - 1$ ]	Target tick size.
<b>-th</b>	Integer: {0, 1} Default: 1	Tick human readable: 0=false, 1=true. If it is true, the sizes on axes are printed in the format 1K, 2M, etc. Note that here, 1K is equivalent to 1000 and not 1024, and so on.
<b>-m</b>	Integer: [1, $2^{32} - 1$ ] Default: 1	Minimum block size. Only the regions with greater sizes than this value will be illustrated.
<b>-vv</b>	—	Vertical view of the output image.
<b>-nn</b>	—	Do not show normalized relative compression (NRC).
<b>-nr</b>	—	Do not show redundancy (self complexity). Smash++ performs reference-based and reference-free compressions to calculate the NRC and redundancy, respectively. If a user does not tend to show them, he/she can turn them off by "-nn" and "-nr" triggers.
<b>-ni</b>	—	Do not show inverse maps.
<b>-ng</b>	—	Do not show regular (not inverse) maps. Smash++ considers by default both regular and reverse complement maps in its calculations.
<b>-h</b>	—	Usage guide.
<b>-v</b>	—	More information (verbose).
<b>--version</b>		Show version.

## S1.4 Example

This section guides step-by-step employing Smash++ to find and visualize rearrangements in two sample genomic sequences. 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++:

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

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

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

There is a 1000 byte reference sequence, named **ref**, as the following:

```
TCCCGGTCTTTTAGCGGCCAGGGGCTGGGCTGTATATCGAAAAGTAATATCCCTTTATGCACCGACCGTAATTATGGACAGCAC
ATATACATTATGAGATTTAAAGATCGCGTGGACGACCACGCGGGCTTATAGCCTCACCTGAGGAAGGGGGTGCCTGCGAGGGAGC
TTGAACCTGTAGCCCCAATCTCGAACGACCTGAGGCTTGTGTGGTCAGAGTTGTGACCAGAGCGATCCCGTTGTCAAATCAACCT
AGAGGAGAGGTAAGGGATACGGGTACATCTCTCCGCTCAGATTGCTCCTATCGGTAGGAAATATCGGGGATAACCCAATACAAA
ACGCTGAAGTGTTCATATTTAGCAAGAAGGGGGGACCGAGGAGCTAAATCAGGGACTATGTAAAATTAGAGTTCTAAGGATAGTA
GCACCCGCGCATGGATCGAACTCCGCCTTGGTTGGGTGTGGATGCCATTGGCGTCACCGTGCTGATAGCCAAATCCCTAGTGAAG
GTAACGTGCGGCCAATTAAATAGTAAGTGGGCGTAGATTCTAGCGGTGGCCTCATACGGCACCAATGTCCATCCTCCCTGTGCCT
AACACTATAACTCCAATCCCGTGAGTCTACCATCGCCGAGAATCAGACGGGCAACAAACCCAGGGAGCGAGAGCTACGCGGCTTA
TATCACGGATGCTTCCCACTAGGGAAATTTGTTCCGCCTAATCCCGTCTTCGCTGGTCAGGCCCGGTCTAGCGGATTACTTC
GCCGGTAATGCAGTACAGAATAAATGAAATTCATTAAGAGAATGAAAGGTTATCAGCGAAGCCTTAAGGTCCAACAAGACGTGCA
TATTCTGAAGTAAACTGGTTGACATGTGTGTAACATGAAGCAGCGCTTATTGATATTATCCGCAACCCACGGGTGGCGGGAAT
CAACCGGCGTCCAGTTCGAACAAGACAGTGGCTACGCATTACGAAATAGGCTTCGTGTTGCTGT
```

and a 1000 byte target sequence, named `tar`, in this directory:

```
CAGCAACACGAAGCCTATTTTCGTAATGCGTAGCGCACTGTCTTGTTCGAACTGGACGCCGGTTGATTCCCGCCAGCCGTGGGGTT
GCGGATAATATCAATAAGCGCGTGCTTCATGTTACACACATGTCAACCAGTTTACTTCAGAATATCGACGTCTTGTGGACCTTA
AGGCTTCGCTGATAACCTTTTCTTCTTAATGAATTTTCTTATTTCTGTACTGCATTACCGCGGAAGTAATCCGCTAGACCGGG
CCTGACCAGCGAAGGACGGGATTAGGCGGAACAAATTTCCCTAGGTGGGGAAGCATCCGTGATATAAGCCGCGTAGCTCTCGCTC
CCTGGGTTTGTGGCCGCTGATTCTCGGCGATGGTAGACTACGGGATTGGAGTTATAGTGTTAGGCACAGGGAGGATGGACAT
TGGTGCCGTATGAGGCCACCGCTAGAATCTACGCCCAGTTACTATTTAATTGGCCGCACGTTACCTTCTAGGATCCCGGTCTT
TTAGCGGCCAGGGGCTGGGCTGTATATCGAAAAGTAATATCCCTTTATGCACCGACCGTAATTATGGACAGCACATATACATTA
TGAGATTTAAAGATCGCGTGGACGACCACGCGGGCTTATAGCCTCACCTGAGGAAGGGGGGCTGCGAGGGAGCTTGAACCTGT
AGCCCCAATCTCGAACGACCTGAGGCTTGTGTGGTCAGAGTGGTGACCAGAGCGATCCCGTTGTCAAATCAACCTAGAGGAGAGG
TAAGGGATACGGGTTACATCTCTCCGCTCAGATTGCTCCTATCGGTAGGAAATATCGGGGATAACCCAATACAAAACGCTGAAGT
GTTTCATATTTAGTAAGAAGCGGTGACCGAGGAGCTAAATCAGGGACTATGTAAAATTAGAGATCTAAGGATAGTAGCACCGCGC
ATGGATCGAACTCCGCATGGTTTGGTGTGATGCCATTGGCGTCACCGTGCTGATAGCCAAATC
```

Running

```
1 ./smashpp -d 1 -dp -fs S -l 3 -r ref -t tar
2 ./smashpp -viz -vv -o example.svg ref.tar.pos
```

results in Fig. S2.

**Fig. S2.** An example of running Smash++ on two 1000 base sequences. (a) the position file and (b) output of the visualizer. One similar region in regular mode and another similar region in inverted mode are detected.

## 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. Schaffer, 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.