Note S1. Methods

Note S1 Methods

The schema of the proposed method is illustrated in Fig. S1. There are five major stages in this method, including

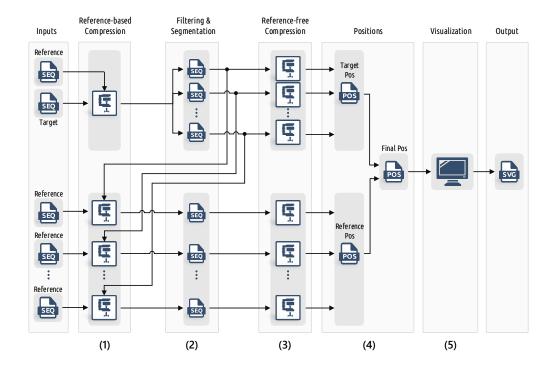


Fig. S1. The schema of Smash++.

S1.1 Building models of the data

S1.2 finding similar regions

In order to smooth the profile information, we use Hann window [1], which is a discrete window function given by

$$w[n] = 0.5 - 0.5 \cos\left(\frac{2\pi n}{N}\right) = \sin^2\left(\frac{\pi n}{N}\right),$$
 (Eq. S1)

in which, $0 \leq n \leq N$ and length of the window is N+1 (Fig. S2).

S1.3 Computing complexities

S1.4 The software

Besides Hann window, that is used as default to filter the profile information obtained by the reference-based compression, we have implemented several other window functions (Fig. S3), including Blackman [1], Hamming [2], Nuttall [3], rectangular [4], sine [5], triangular [6] and Welch [7]

S1.4. The software

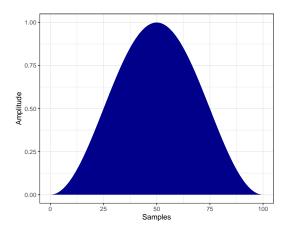


Fig. S2. Hann window for 101 samples.

windows. These functions are given by

$$\begin{split} w[n] &= 1, & \text{(rectangular)} \\ w[n] &= 1 - \left| \frac{n - N/2}{L/2} \right|, \quad L = N, & \text{(triangular/Bartlett)} \\ w[n] &= 1 - \left(\frac{n - N/2}{N/2} \right)^2, & \text{(Welch)} \\ w[n] &= \sin\left(\frac{\pi n}{N}\right), & \text{(sine)} \\ w[n] &= 0.54348 - 0.45652 \cos\left(\frac{2\pi n}{N}\right), & \text{(Hamming)} \\ w[n] &= 0.42659 - 0.49656 \cos\left(\frac{2\pi n}{N}\right) + 0.07685 \cos\left(\frac{4\pi n}{N}\right), & \text{(Blackman)} \\ w[n] &= 0.35577 - 0.48740 \cos\left(\frac{2\pi n}{N}\right) + 0.14423 \cos\left(\frac{4\pi n}{N}\right) - 0.01260 \cos\left(\frac{6\pi n}{N}\right). & \text{(Nuttall)} \\ &\text{(Eq. S2)} \end{split}$$

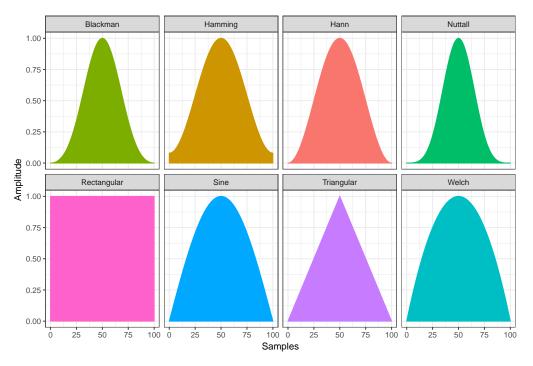


Fig. S3. Window functions.

Note S2 Experiment setup

S2.1 Datasets

Table S1. Datasets used in the experiments.

Category	Reference	Length (base)	Target	Length (base)	Description
Synthetic	RefS	1,000	TarS	1,000	
Synthetic	RefM	100,000	TarM	100,000	
Synthetic	RefL	5,000,000	TarL	5,000,000	
Synthetic	RefXL	100,000,000	TarXL	100,000,000	

Note S3. Results 4

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 [8]. 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++.

S4.1 Install

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

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

make
```

S4.2 Run

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

```
1 ./smashpp
```

provides the following guide:

```
SYNOPSIS
2
     ./smashpp OPTIONS... -r REF-FILE -t TAR-FILE
3
4
   SAMPLE
5
   DESCRIPTION
6
7
     Mandatory arguments
     -r, --ref FILE
8
                                 reference file (Seq/Fasta/Fastq)
     -t, --tar FILE
                                                (Seq/Fasta/Fastq)
9
                                 target file
10
11
     Options
     -v, --verbose
                                 more information
12
     -1, --level INT
                                level of compression: [0, 5]
13
     -m, --min INT
                                 min segment size: [1, 4294967295]
14
15
     -nr, --no-redun
                                 do NOT compute self complexity
                                 Entropy of 'N's: [0.0, 100.0]
     -e, --ent-n FLOAT
16
     -n, --nthr INT
                                 number of threads: [1, 8]
17
     -fs, --filter-scale S\,|\,M\,|\,L\, scale of the filter:
18
19
                                 {S|small, M|medium, L|large}
                                 window size: [1, 4294967295]
     -w. --wsize INT
20
     -wt, --wtype INT/STRING
                                 type of windowing function:
21
22
                                 {O|rectangular, 1|hamming, 2|hann,
23
                                 3|blackman, 4|triangular, 5|welch,
24
                                 6|sine, 7|nuttall}
     -d, --step
25
                   INT
                                 sampling steps
     -th, --thresh FLOAT
26
                                 threshold: [0.0, 20.0]
27
     -sp, --save-profile
                                 save profile (*.prf)
28
     -sf, --save-filter
                                 save filtered file (*.fil)
     -sb, --save-seq
29
                                 save sequence (input: Fasta/Fastq)
30
     -ss, --save-segment
                                 save segmented files (*-s_i)
     -sa, --save-all
31
                                 save profile, filetered and
32
                                 segmented files
     -h, --help
                                 usage guide
33
34
     -rm, --ref-model k,[w,d,] ir, a,g/t, ir, a,g:...
35
     -tm, --tar-model k, [w,d,]ir,a,g/t,ir,a,g:...
                                 parameters of models
36
                           (INT) k: context size
37
38
                           (INT) w: width of sketch in log2 form,
                                     e.g., set 10 for w=2^{-1}0=1024
39
                           (INT) d: depth of sketch
40
41
                           (INT) ir: inverted repeat: {0, 1, 2}
```

S4.3. Example 6

```
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 [9]. 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 S1.4, 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".

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 7

References

[1] 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.

- [2] J. W. Tukey and R. W. Hamming, Measuring noise color. Bell Telephone Laboratories, 1949.
- [3] 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.
- [4] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 1999.
- [5] 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.
- [6] M. S. Bartlett, "Periodogram analysis and continuous spectra," Biometrika, vol. 37, no. 1/2, pp. 1–16, 1950.
- [7] 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.
- [8] M. Hosseini, D. Pratas, and A. J. Pinho. Smash++. [Online]. Available: https://github.com/smortezah/smashpp
- [9] D. Pratas. Goose. [Online]. Available: https://github.com/pratas/goose