Supplementary Material for

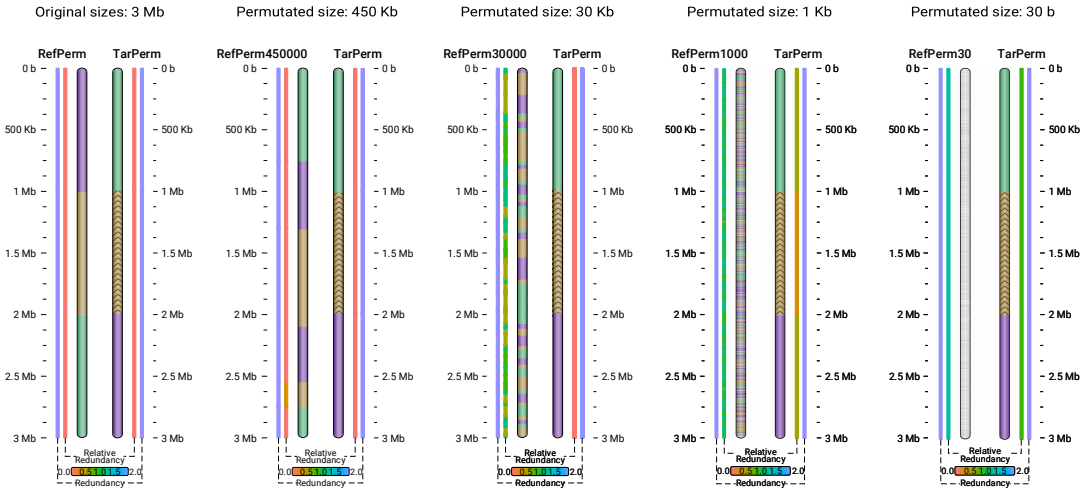
Smash++: finding rearrangements

Morteza Hosseini[1], Diogo Pratas[1,2], Armando J. Pinho[1]

[1]IEETA/DETI, University of Aveiro, Portugal, [2]Dept. of Virology, University of Helsinki, Finland
{seyedmorteza,pratas,ap}@ua.pt

Note S1 Permutation

and ??.



Note S2 Smash++ compared with other methods

The results of running Smash++ and other methods on real dataset is shown in Figures S1, S2, S3 and S4.

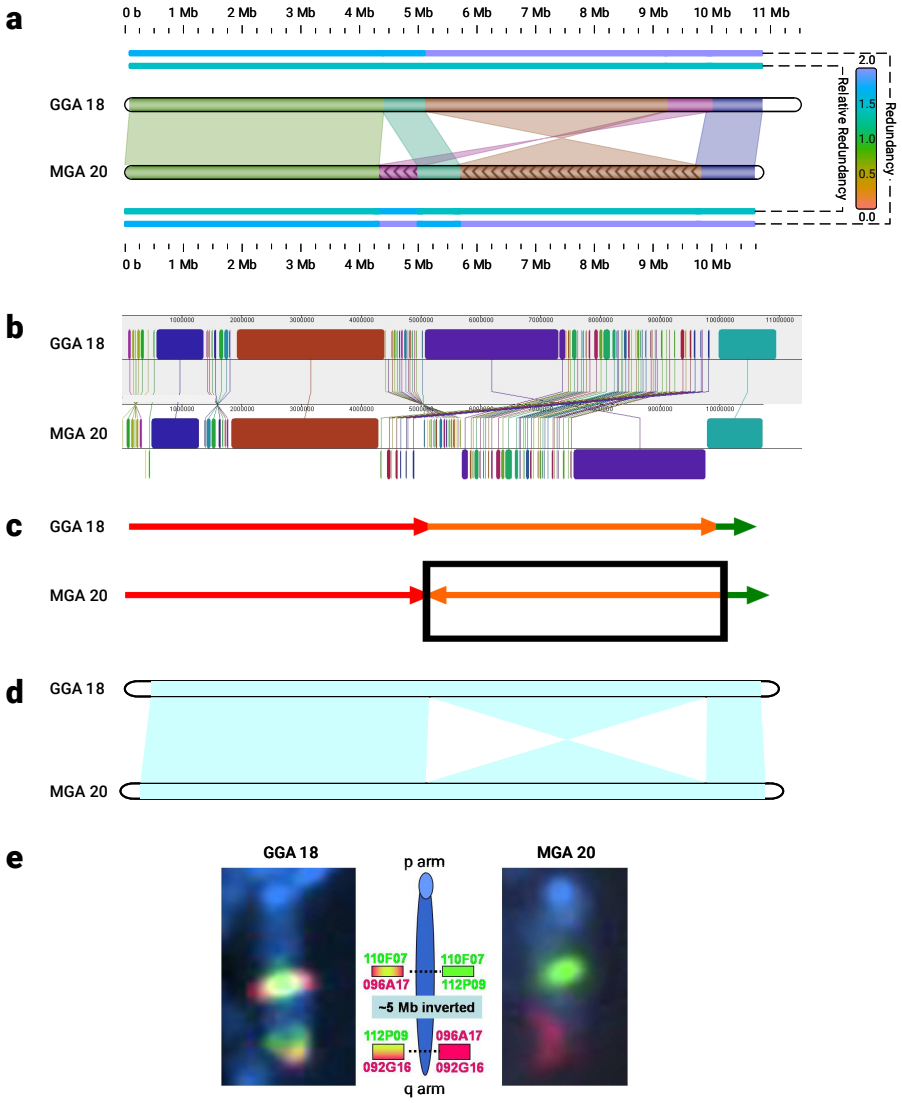


Fig. S1. Pair-wise comparison of *G. gallus* chromosome 18 and *M. gallopavo* chromosome 20. (a) Smash++, with $k = 14$ and 5 used by an FCM and an STMM, respectively. The blocks smaller than 500 Kb are discarded; (b) progressiveMauve [1], with LCB (locally collinear block) weight of 18692. Reverse complements are shown in lower level; (c) adopted from [2], which is confirmed by fluorescence *in situ hybridization* (FISH) analysis. The box shows a local rearrangement; (d) SynBrowser [3], with the resolution of 150 Kb (minimum size of a reference block); (e) adopted from [4], which confirms an inversion rearrangement of size ∼5 Mb by FISH analysis.

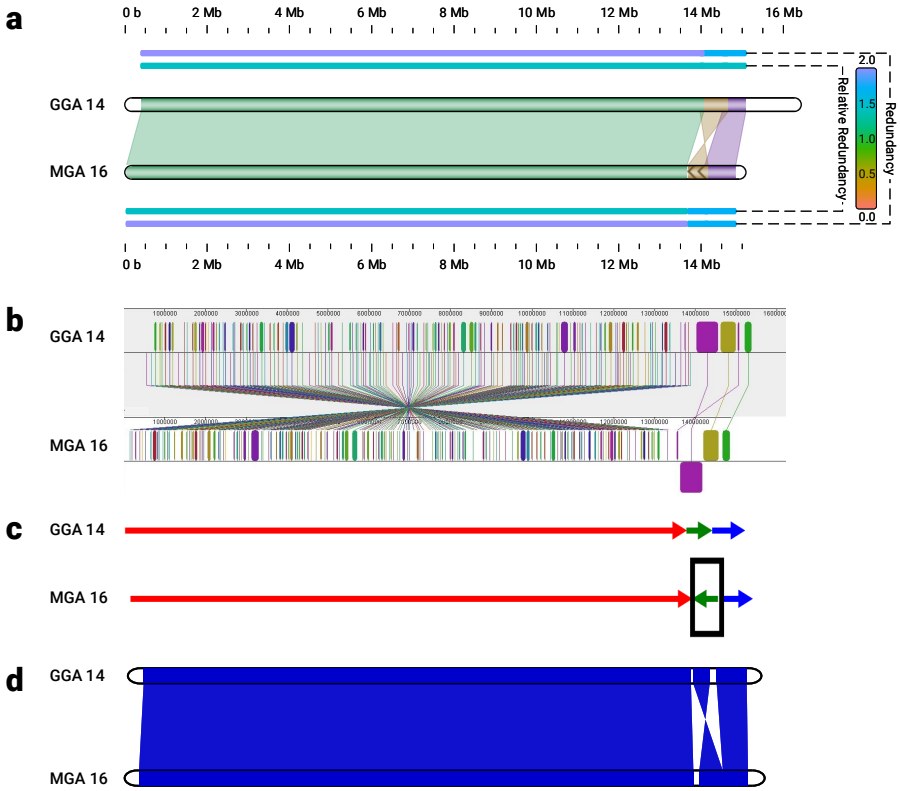


Fig. S2. *G. gallus* chromosome 14 compared to *M. gallopavo* chromosome 16. (a) Smash++, employing an FCM and an STMM with $k = 14$ and 5, respectively. The blocks smaller than 400 Kb are discarded; (b) progressiveMauve, with LCB weight of 27424; (c) adopted from [2]. The box shows an inversion rearrangement; (d) SynBrowser, with the resolution of 150 Kb.

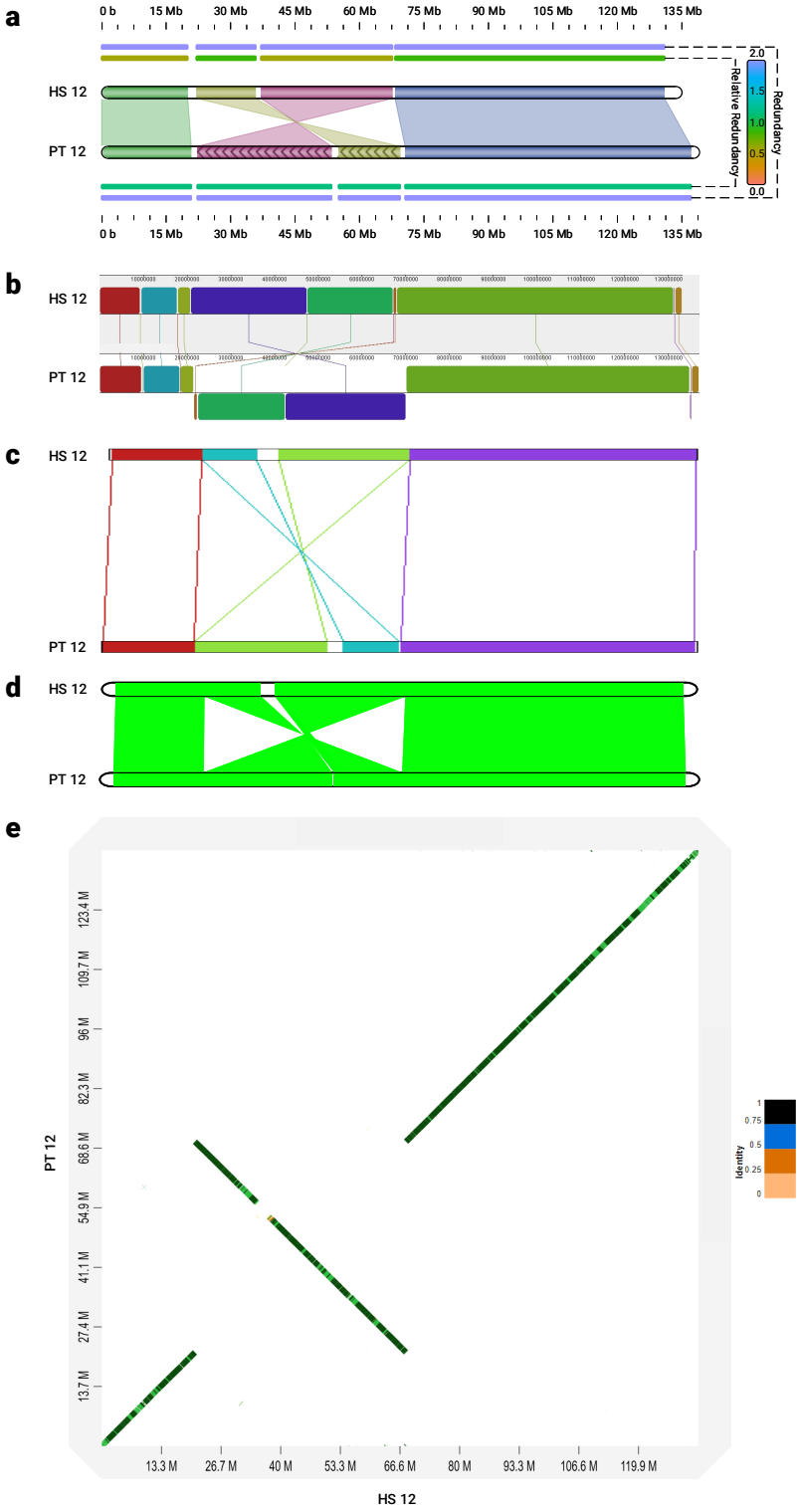


Fig. S3. Comparison of *H. sapiens* chromosome 12 and *P. troglodytes* chromosome 12. (a) Smash++, with $k = 14$ used by an FCM. The blocks smaller than 100 Kb are discarded; (b) progressiveMauve, with LCB weight of 55186; (c) Cinteny [5], with minimum length of synteny block = 1 Kb, maximum gap between adjacent markers = 5 Mb and minimum number of markers = 1; (d) SynBrowser, with the resolution of 150 Kb; (e) D-Genies [6], in "strong precision" mode.

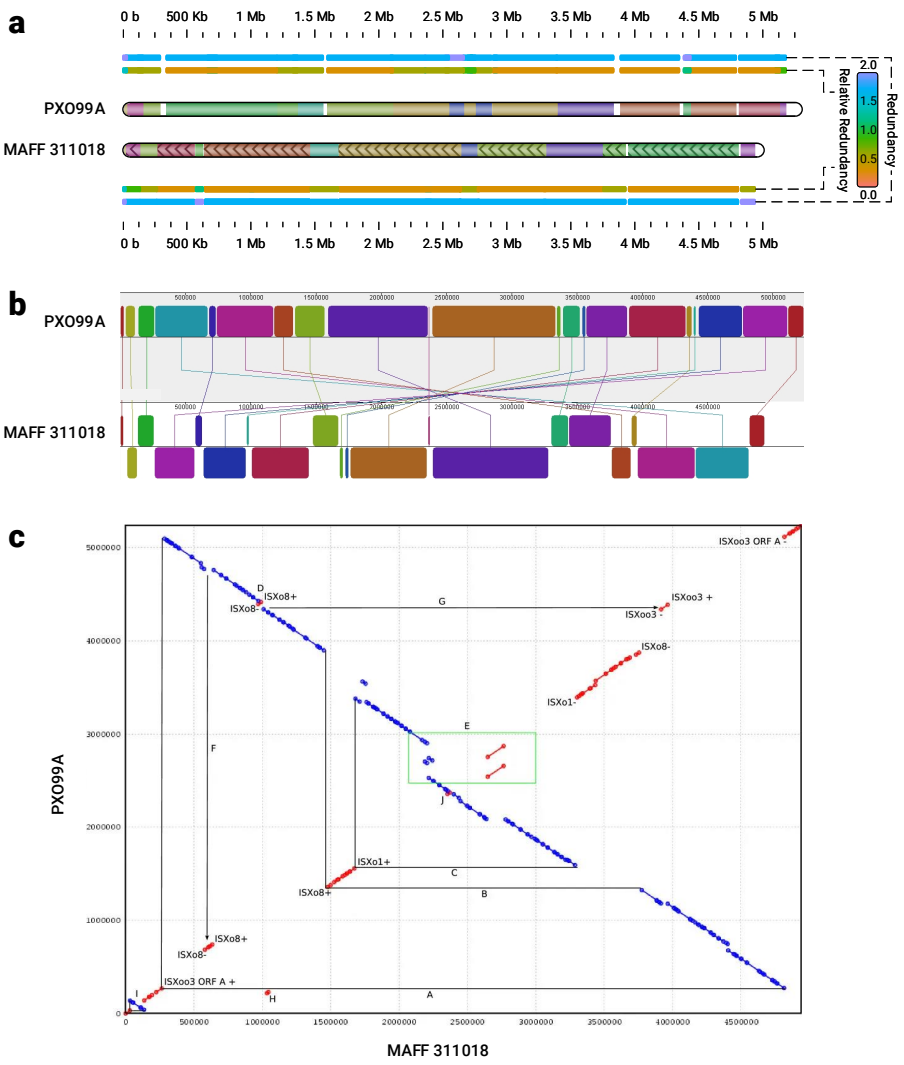


Fig. S4. Pair-wise comparison of PXO99A and MAFF 311018. (a) Smash++, with k=13 used by an FCM. The blocks smaller than 10 Kb are discarded. In order to make the figure clearer, the shaded paths for connecting corresponding regions are not drawn; (b) progressiveMauve, with LCB weight of 3926; (c) adopted from [7], which employs an alignment-based method to obtain this dot plot. The blue and red colors shows regions of PXO99A that align to the same or opposite strand of MAFF 311018, respectively.

Note S3 Software manual

Smash++ is implemented in C++ language and is publicly available at [8], 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.

S3.1 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

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

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
```

S3.2 Run

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:

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

Table S1. Options provided by Smash++ interface.

Flag	Input	Description
Required		
`-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.
Optional		
`-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 [9].
`-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 [10], Hamming [11], Nuttall [12], rectangular [13], sine [14], triangular [15] and Welch [16] windows. These functions along with Hann function are given by Equation S1 and are plotted in Fig. S5.
`-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: $[1, 2^{64} - 1]$	Sampling steps. Instead of considering the whole information content, we can make samples of it. The default value is $\lceil \min(\lvert ref \rvert, \lvert tar \rvert) / 5000 \rceil$. Therefore, if this flag is not entered by user, a maximum number of 5000 values of the information content will be considered.
`-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: $[-2^{15}, 2^{15}-1]$ Default: 0	Target ending guard. 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, by resizing 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 altered. Therefore, changing the guard sizes of target regions will affect the sizes of reference regions. The same thing happens for the sizes of target regions, when the guard sizes of reference regions are changed.
`-ar`	N/A	Consider asymmetric regions. It makes Smash++ not to balance the similar reference and target regions.
`-nr`	N/A	Do not compute self complexity. It makes the tool not to perform the reference-free compression (self-complexity computation).
`-sb`	N/A	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`	N/A	Save profile, *.prf. When the information profile is obtained, Smash++ smoothens then removes it, by default; however, it can be preserved by this option.
`-sf`	N/A	Save filtered file, *.fil. The filtered profile is segmented then removed, by default; however, it can be preserved by this option.
`-ss`	N/A	Save segmented files, *.s_i.
`-sa`	N/A	Save all generated files, including profile, filtered and segmented files.
`-rm`	k,[w,d,]ir,α,γ/t,ir,α,γ:...	Parameters of reference models.
`-tm`	k,[w,d,]ir,α,γ/t,ir,α,γ:... Default: $14, 0, 0.001, 0.95/5, 0, 0.001, 0.95$	Parameters of target models: • k (integer > 1): context size, • w (integer $< 2^{64} - 1$): Count-Min-Log sketch width in $\log_2$ form, e.g., set 10 for $w = 2^{10} = 1024$, • d (integer > 0): Count-Min-Log sketch depth, • ir (integer: {0, 1, 2}): inverted repeat, including 0: regular (not inverted), 1: inverted solely, and 2: both regular and inverted, • α (float > 0): estimator, • γ (float: $[0.0, 1.0)$): forgetting factor, • t (integer > 0): threshold (number of substitutions). 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.
`-ll`	N/A	List of compression levels. It shows the list of parameters that would be chosen automatically for each model.
`-h`	N/A	Usage guide.
`-v`	N/A	More information (verbose).
`--version`		Show the version.

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

(Eq. S1)

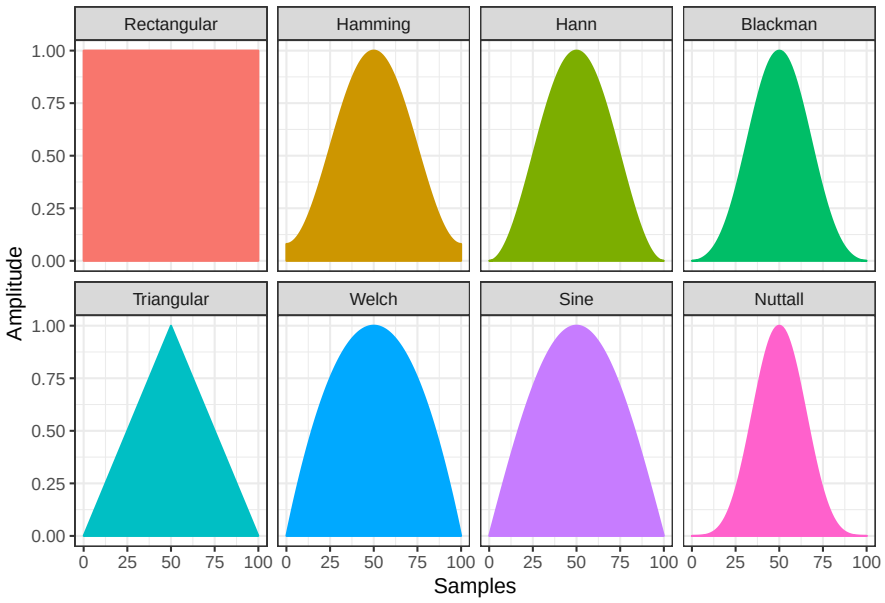


Fig. S5. Various windowing functions embedded in Smash++.

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 (Invereted 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.

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

S3.3 Run visualizer

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

```
./smashpp -viz
```

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

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

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

Flag	Input	Description
Required		
	*.pos file	Position file, generated by Smash++ tool. Spaces in the name is supported. It can be redirected to Smash++ by standard input ("stdin").
Optional		
`-o`	*.svg file Default: map.svg	Output image name.
`-rn`	String	Reference name shown on output.
`-tn`	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".
`-l`	Integer: $[1, 6]$ Default: 1	Type of the link between similar regions.
`-c`	Integer: $[0, 1]$ Default: 0	Color mode.

`-p`	Float: $[0.0, 1.0]$ Default: 0.9	Opacity.
`-w`	Integer: $[8, 100]$ Default: 10	Width of the sequence.
`-s`	Integer: $[5, 200]$ Default: 40	Space between sequences.
`-tc`	Integer: $[1, 255]$	Total number of colors in the map, which is automatically chosen by default.
`-rt`	Integer: $[1, 2^{32}-1]$	Reference tick size.
`-tt`	Integer: $[1, 2^{32}-1]$	Target tick size.
`-th`	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.
`-m`	Integer: $[1, 2^{32}-1]$ Default: 1	Minimum block size. Only the regions with greater sizes than this value will be illustrated.
`-vv`	N/A	Vertical view of the output image.
`-nrr`	N/A	Do not show relative redundancy (relative complexity).
`-nr`	N/A	Do not show redundancy (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.
`-ni`	N/A	Do not show inverse maps.
`-ng`	N/A	Do not show regular (not inverse) maps. Smash++ considers by default both regular and reverse complement maps in its calculations.
`-h`	N/A	Usage guide.
`-v`	N/A	More information (verbose).
`--version`		Show version.

S3.4 Example

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

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 "ref", as the following:

```
TCCCGGTCTTTTAGCGGCCAGGGGCCTGGGCTGTATATCGAAAAGTAATATCCCTTTATGCACCGACCGTAATTATGGACAGCAC
ATATACATTATGAGATTTAAAGATCGCGTGGACGACCACGCGGGCTTATAGCCTCACCTGAGGAAGGGGGTGCCTGCGAGGGAGC
TTGAACCTGTAGCCCCAATCTCGAACGACCTGAGGCTTGTGTGGTCAGAGTTGTGACCAGAGCGATCCCGTTGTCAAATCAACCT
AGAGGAGAGGTAAGGGATACGGGTTACATCTCTCCGCTCAGATTGCTCCTATCGGTAGGAAATATCGGGGATAACCCAATACAAA
ACGCTGAACTGTTCATATTTAGCAAGAAGGGGGGACCGAGGAGCTAAATCAGGGACTATGTAAAATTAGAGTTCTAAGGATAGTA
GCACCCGCGCATGGATCGAACTCCGCCTTGGTTGGGTGTGGATGCCATTGGCGTCACCGTGCTGATAGCCAAATCCCTAGTGAAG
GTAACGTGCGGCCAATTAAATAGTAACTGGGCGTAGATTCTAGCGGTGGCCTCATACGGCACCAATGTCCATCCTCCCTGTGCCT
AACACTATAACTCCAATCCCGTGAGTCTACCATCGCCGAGAATCAGACGGGCAACAAACCCAGGGAGCGAGAGCTACGCGGCTTA
TATCACGGATGCTTCCCCACCTAGGGAAATTTGTTCCGCCTAATCCCGTCCTTCGCTGGTCAGGCCCGGTCTAGCGGATTACTTC
GCCGGTAATGCAGTACAGAATAAATGAAATTCATTAAGAGAATGAAAGGTTATCAGCGAAGCCTTAAGGTCCAACAAGACGTCGA
TATTCTGAAGTAAACTGGTTGACATGTGTGTAACATGAAGCACGCGCTTATTGATATTATCCGCAACCCCACGGCTGGCGGGAAT
CAACCGGCGTCCAGTTCGAACAAGACAGTGCGCTACGCATTACGAAATAGGCTTCGTGTTGCTGT
```

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

```
CAGCAACACGAAGCCTATTTCGTAATGCGTAGCGCACTGTCTTGTTCGAACTGGACGCCGGTTGATTCCCGCCAGCCGTGGGGTT
GCGGATAATATCAATAAGCGCGTGCTTCATGTTACACACATGTCAACCAGTTTACTTCAGAATATCGACGTCTTGTTGGACCTTA
AGGCTTCGCTGATAACCTTTCATTCTCTTAATGAATTTCATTTATTCTGTACTGCATTACCGGCGAAGTAATCCGCTAGACCGGG
CCTGACCAGCGAAGGACGGGATTAGGCGGAACAAATTTCCCTAGGTGGGGAAGCATCCGTGATATAAGCCGCGTAGCTCTCGCTC
CCTGGGTTTGTTGCCCGTCTGATTCTCGGCGATGGTAGACTCACGGGATTGGAGTTATAGTGTTAGGCACAGGGAGGATGGACAT
TGGTGCCGTATGAGGCCACCGCTAGAATCTACGCCCAGTTACTATTTAATTGGCCGCACGTTACCTTCACTAGGATCCCGGTCTT
TTAGCGGCCAGGGGCCTGGGCTGTATATCGAAAAGTAATATCCCTTTATGCACCGACCGTAATTATGGACAGCACATATACATTA
TGAGATTTAAAGATCGCGTGGACGACCACGCGGGCTTATAGCCTCACCTGAGGAAGGGGGGGCCTGCGAGGGAGCTTGAACCTGT
AGCCCCAATCTCGAACGACCTGAGGCTTGTGTGGTCAGAGTGGTGACCAGAGCGATCCCGTTGTCAAATCAACCTAGAGGAGAGG
TAAGGGATACGGGTTACATCTCTCCGCTCAGATTGCTCCTATCGGTAGGAAATATCGGGGATAACCCAATACAAAACGCTGAACT
GTTCATATTTAGTAAGAACGGGTGACCGAGGAGCTAAATCAGGGACTATGTAAAATTAGAGATCTAAGGATAGTAGCACCCGCGC
ATGGATCGAACTCCGCCATGGTTTGGTGTCGATGCCATTGGCGTCACCGTGCTGATAGCCAAATC
```

Next, run Smash++ and the visualizer:

```
./smashpp -r ref -t tar
./smashpp -viz -o example.svg ref.tar.pos
```

Results are shown in Fig. S6. Note that the position file is saved as “ref.tar.pos” and the output plot is saved as “example.svg”.

a

```
##SMASH++
##PARAM=<-r ref -t tar>
##INFO=<Ref=ref,RefSize=1000,Tar=tar,TarSize=1000>
#RBeg   REnd   RRelRdn   RRdn     TBeg   TEnd   TRelRdn   TRdn     Inv
0       510    0.5108    2.0000   497    1000   0.5120    2.0000   F
498     1000   0.1518    2.0000   508    0      0.1551    2.0000   T
```

b

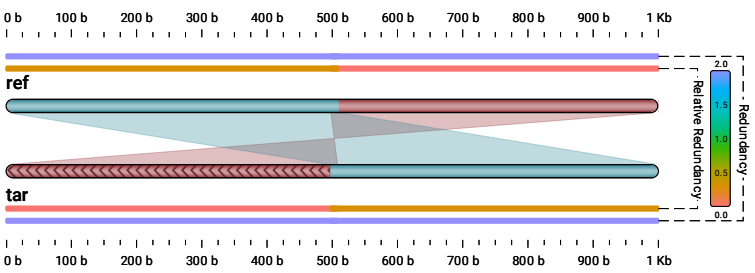


Fig. S6. An example of running Smash++ on two 1000 base sequences. (a) the position file and (b) output of the visualizer. One similar region in regular mode and another one in inverted mode are detected.

References

[1] A. E. Darling, B. Mau, and N. T. Perna, "progressivemauve: multiple genome alignment with gene gain, loss and rearrangement," *PloS one*, vol. 5, no. 6, p. e11147, 2010.

[2] Y. Zhang, X. Zhang, T. H. O'Hare, W. S. Payne, J. J. Dong, C. F. Scheuring, M. Zhang, J. J. Huang, M.-K. Lee, M. E. Delany *et al.*, "A comparative physical map reveals the pattern of chromosomal evolution between the turkey (*Meleagris gallopavo*) and chicken (*Gallus gallus*) genomes," *BMC genomics*, vol. 12, no. 1, p. 447, 2011.

[3] J. Lee, W.-y. Hong, M. Cho, M. Sim, D. Lee, Y. Ko, and J. Kim, "Synteny portal: a web-based application portal for synteny block analysis," *Nucleic acids research*, vol. 44, no. W1, pp. W35–W40, 2016.

[4] R. A. Dalloul, J. A. Long, A. V. Zimin, L. Aslam, K. Beal, L. A. Blomberg, P. Bouffard, D. W. Burt, O. Crasta, R. P. Crooijmans *et al.*, "Multi-platform next-generation sequencing of the domestic turkey (*Meleagris gallopavo*): genome assembly and analysis," *PLoS biology*, vol. 8, no. 9, p. e1000475, 2010.

[5] A. U. Sinha and J. Meller, "Cinteny: flexible analysis and visualization of synteny and genome rearrangements in multiple organisms," *BMC bioinformatics*, vol. 8, no. 1, p. 82, 2007.

[6] F. Cabanettes and C. Klopp, "D-genies: dot plot large genomes in an interactive, efficient and simple way," *PeerJ*, vol. 6, p. e4958, 2018.

[7] S. L. Salzberg, D. D. Sommer, M. C. Schatz, A. M. Phillippy, P. D. Rabinowicz, S. Tsuge, A. Furutani, H. Ochiai, A. L. Delcher, D. Kelley *et al.*, "Genome sequence and rapid evolution of the rice pathogen *Xanthomonas oryzae* pv. *oryzae* PXO99A," *BMC genomics*, vol. 9, no. 1, p. 204, 2008.

[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

[10] 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.

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

[12] 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.

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

[14] 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.

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

[16] 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.