Hapo-G, haplotype-aware polishing of genome assemblies with accurate reads

Jean-Marc Aury, Benjamin Istace

Published online 3 May 2021

NAR Genomics and Bioinformatics, 2021, Vol. 3, No. 2 **1**
doi: 10.1093/nargab/lqab034

Hapo-G, haplotype-aware polishing of genome assemblies with accurate reads

Jean-Marc Aury* **and Benjamin Istace**

Génomique Métabolique, Genoscope, Institut François Jacob, CEA, CNRS, Univ Evry, Université Paris-Saclay, 91057 Evry, France

Received December 21, 2020; Revised March 18, 2021; Editorial Decision March 30, 2021; Accepted April 13, 2021

ABSTRACT

Single-molecule sequencing technologies have recently been commercialized by Pacific Biosciences and Oxford Nanopore with the promise of sequencing long DNA fragments (kilobases to megabases order) and then, using efficient algorithms, provide high quality assemblies in terms of contiguity and completeness of repetitive regions. However, the error rate of long-read technologies is higher than that of short-read technologies. This has a direct consequence on the base quality of genome assemblies, particularly in coding regions where sequencing errors can disrupt the coding frame of genes. In the case of diploid genomes, the consensus of a given gene can be a mixture between the two haplotypes and can lead to premature stop codons. Several methods have been developed to polish genome assemblies using short reads and generally, they inspect the nucleotide one by one, and provide a correction for each nucleotide of the input assembly. As a result, these algorithms are not able to properly process diploid genomes and they typically switch from one haplotype to another. Herein we proposed Hapo-G (Haplotype-Aware Polishing Of Genomes), a new algorithm capable of incorporating phasing information from high-quality reads (short or long-reads) to polish genome assemblies and in particular assemblies of diploid and heterozygous genomes.

INTRODUCTION

Long-read technologies commercialized by Pacific Biosciences (PACBIO) and Oxford Nanopore Technologies (ONT) are able to sequence long DNA molecules but at the cost of a higher error rate at least for standard protocols. Their throughputs are sufficient to generate complex genomes (1–5) and their costs are almost compatible with their use in large-scale resequencing projects (6–8). Standard genome assemblies currently rely on a combination of several technologies, making it possible to generate complete assemblies in terms of both repetitive and coding regions. The quality of the consensus relies heavily on the use of high-quality reads, mainly short reads, and the choice of a polishing algorithm.

One of the most popular polishing algorithms, Pilon (9), was developed several years ago, before the advent of the long-read era and was originally designed to detect variants and improve microbial genome assemblies. With the increasing popularity of long-read technologies, public databases now contain a large collection of very contiguous assemblies, but even if the overall reported quality seems sufficient, local errors can critically affect protein prediction (10). Aware of this issue, the bioinformatic community has developed several tools over the past 2 years (11–16). Most of the tools (Pilon, Racon, NextPolish, HyPo, Apollo and POLCA) are based on short-read alignment, ntEdit is the only method that uses a kmer approach and NextPolish combines both strategies (Table 1). After aligning the short reads, the algorithms detect errors by examining the pileup of bases from the reads (Pilon, NextPolish), by generating a consensus using Partial Order Alignment (Racon, HyPo) or by detecting variants (POLCA). While these tools are capable of correcting most of the errors in a draft assembly generated using long reads, we have observed frequent issues when correcting heterozygous regions. Indeed, the case of diploid genomes is particularly problematic since in this case the long-read assembly is composed of collapsed homozygous regions and duplicated allelic regions which will complicate the correct alignment of short reads. As the existing tools work locally and not at the scale of a 150 bp read and its mate, they frequently generate a mixture of haplotypes. Switching between haplotypes is problematic for the alignment of short reads and variant calling, but it can also affect the coding sequence of genes. As an example, when we were dealing with a long-read genome assembly, we observed that the Pilon correction was not able to restore a deletion in a heterozygous coding region (Figure 1). This simple observation motivated our need to develop a new polishing algorithm.

Here we present Hapo-G (pronounced as apogee), a new method dedicated to the polishing of genome assemblies.

*To whom correspondence should be addressed. Tel: +33 1 60 87 36 03; Email: jmaury@genoscope.cns.fr

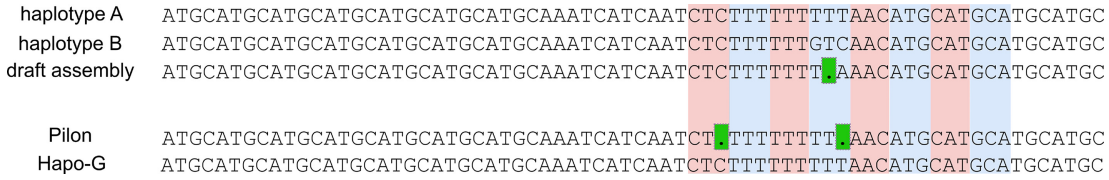

Figure 1. Example of a deletion in a coding frame. The two haplotypes have consistent coding frames (codons are alternatively colored in red and blue) and the draft assembly contains a deletion in a stretch of T's (green box). Pilon was not able to restore the coding frame and add a second frameshift. In comparison Hapo-G was able to restore haplotype A.

Table 1. General characteristics of existing polishing algorithms. These seven tools were evaluated in our benchmark with the specified parameters

	Reference	Publication date	Version	Read alignment	Integrated aligner	Parameters used in the benchmark
Hapo-G	This study	-	0.1	yes	BWA	-t 36
Apollo	Firtina C. *et al.* (12)	2020	2.0	yes	no	-t 36
POLCA	Ziminn AV. *et al.* (14)	2020	3.4.2	yes	BWA	-t 36
HyPo	Kundu R. *et al.* (13)	2019	1.0.3	yes	no	-s size* -c 180 -t 36
NextPolish	Hu J. *et al.* (12)	2019	1.3.2	yes	BWA	task = best parallel_jobs = 6 multithread_jobs = 6 genome_size = auto
ntEdit	Warren RL. *et al.* (11)	2019	1.3.2	no	NA	-k 40 -t 36 –outbloom –solid ntEdit: -m 1 -t 36
Racon	Vaser R. *et al.* (15)	2017	1.4.3	yes	no	-t 36
Pilon	Walker BJ. *et al.* (9)	2014	1.23	yes	no	–threads 36

*size = 120Mb (*Arabidopsis thaliana*), 120Mb (*Solanum tuberosum*), 100 K (synthetic sequence).

This algorithm tends to phase the assembly while correcting the sequencing errors. We compare Hapo-G with existing polishers (HyPo, Apollo, NextPolish, Pilon, POLCA and Racon) and show that Hapo-G is not only comparable to existing methods for polishing draft assemblies, but also faster and tends to decrease jumps between haplotypes. Hapo-G is written in C, uses the hts library (17) and is freely available at http://www.genoscope.cns.fr/hapog.

MATERIALS AND METHODS

Hapo-G algorithm

Hapo-G, like most existing tools, requires a sorted bam file containing the high-quality read alignments on the draft genome. These alignments could have been generated using bwa mem (18), minimap2 (19) or any other alignment tool capable of producing a bam file. Hapo-G maintains two stacks of alignments, the first (all-ali) contains all the alignments that overlap the currently inspected base, and the second (hap-ali) contains only the read alignments that agree with the last selected haplotype. Hapo-G selects a reference alignment and tries to use it as long as possible to polish the region where it aligns, which will minimize mixing between haplotypes (Figure 2).

Hapo-G performs the polishing sequentially and scans the input bam file of alignments sorted by position. For each input alignment (called current alignment), Hapo-G polishes each nucleotide in the region between the last recorded position and the start position of the current alignment (called current position), after which the current alignment will be added to both stacks (Figure 2).

Polishing of a given nucleotide in the draft assembly

First, the two stacks (all-ali and hap-ali), if they are not empty, are cleaned to remove any alignment that does not overlap with the current position. In case the reference alignment has been deleted, a new alignment is selected from the hap-ali stack (the read alignment that ends closest to the current position). If the coverage at the current position is below a threshold, set at three reads, the current base in the draft sequence remains unchanged. Otherwise, the nucleotide of the reference alignment (called the reference base) is extracted and the frequency of this reference base is calculated in the all-ali and hap-ali stacks. Based on its frequency, the current position is tagged as a homozygous site, a heterozygous site or a sequencing error (Figure 2).

The position is classified as homozygous if the frequency of the reference base is >0.8 and at least 3 reads from the hap-ali stack are in accordance. If the reference base and the nucleotide of the draft assembly (the current base) are different, the current base is replaced by the reference base.

The position is classified as heterozygous if the frequency of the reference base is between 0.2 and 0.8 and the hap-ali stack contains at least six reads. If the reference base and the current base are different, the current base is replaced by the reference base. In addition, any read alignments that do not have the same base as the reference base at the current position will be removed from the hap-ali stack. Indeed, they may represent a second haplotype. Importantly, when a read is removed from the stack, its name and its mate name are added to a hash table. The corresponding read align-

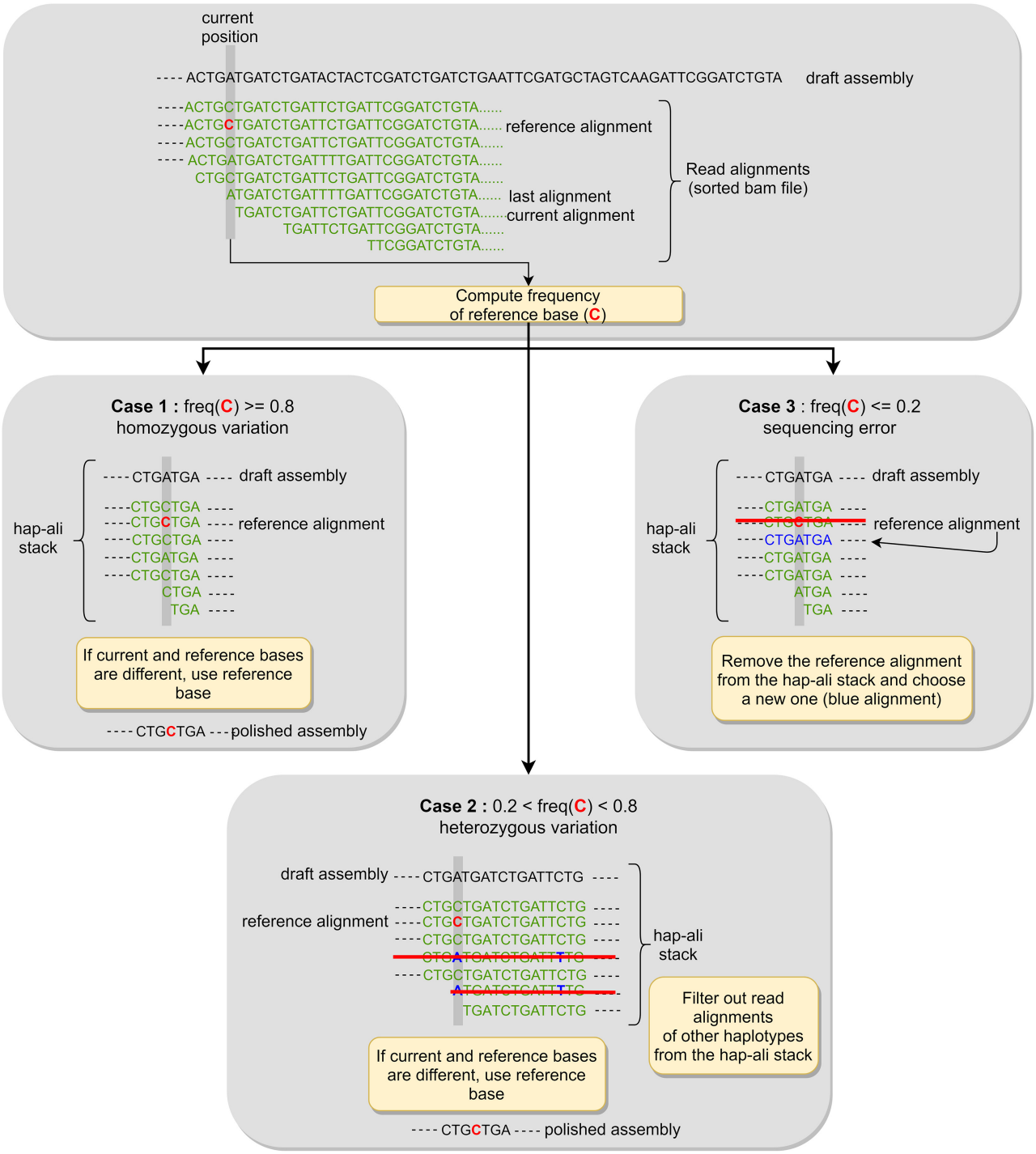

Figure 2. Description of the Hapo-G algorithm. Two stacks of alignments are stored, all-ali which contains all the alignment of a specific region (not shown) and hap-ali which contains reads from the same haplotype. The bam file is processed iteratively, and for each input alignment, Hapo-G will polish the region (draft genome is in black) between the start of the last alignment and the start of the current alignment. The reference alignment is the one used as the backbone for error-correction. Once the frequency of the reference base (in red) is computed, the position is classified as homozygous (case 1, left panel), heterozygous (case 2, lower panel) or sequencing error (case 3, right panel).

ments will be ignored when encountered later while polishing the current sequence. The hash table is empty when the end of the current sequence is reached.

Usage and parallelization of Hapo-G

The polishing step of Hapo-G is wrapped in a python script which manages the pre and post processing steps. First, the wrapper indexes the genome and maps the reads on the draft assembly using bwa-mem for short reads and minimap2 for long-reads. The polishing step, written in C using the htslib, is not multithreaded but can be easily parallelized by splitting the input fasta file as well as the alignment file. This divide and conquer strategy makes it possible to speed up the polishing step and allows to take advantage of a wide range of computing architectures.

Generation of benchmarking datasets

Homozygous genome assemblies. We downloaded the Nanopore data of an *Arabidopsis thaliana* sample produced by Michigan State University (Table 2) and assembled these data using the Flye assembler (20) (v2.8.1) with the parameter '-g 120m' to indicate a genome size of 120 Mb. We obtained an assembly of 118 Mb, comprising 16 large contigs and a contig N50 of 14.8 Mb (Table 2). This assembly was used as input of the Medaka polisher (v1.2.0, https://github.com/nanoporetech/medaka), in conjunction with the Nanopore reads and the '-m r941_min_high_g303' parameter was applied, in order to choose a suitable model for this type of data. This assembly was used to compare polishing tools.

In addition, we downloaded a human genome assembly based on Nanopore reads that was published by Jain *et al*. (2) as well as Illumina reads from the same individual (ERR194147). This dataset was used to benchmark all polishers, in terms of quality of the result but also of restitution time and memory requirements.

Heterozygous genome assemblies. We generated a 100 kb sequence using an Homo sapiens model (http://rsat.sb-roscoff.fr/RSAT_home.cgi) and created two haplotypes by incorporating, each time, 100 random mutations into the initial 100 kb sequence (http://www.bioinformatics.org/sms2/mutate_dna.html). We added two random 1 kb sequences to both ends of the two haplotypes to avoid mapping issues on first and last nucleotides. Illumina short-reads were generated from both haplotypes using ART (21) software (version 2.5.1) and the following parameters: -ss HSXt -p -l 150 -f 25 -m 200 -s 10. From the two haplotype sequences, we generated an haploid sequence by alternatively retaining 60 nucleotides of each sequence and adding 2000 random mutations (http://www.bioinformatics.org/sms2/mutate_dna.html) to simulate sequencing errors. The resulting haploid sequence is a mixture of the two haplotypes and is used to test the ability of each polisher to correct errors and phase the draft sequence.

Additionally, we downloaded the Nanopore data produced by the authors of a recent article by Zhou *et al*. (22) describing the diploid assembly of *Solanum tuberosum* (Table 2). The long-read dataset was assembled using the Flye assembler (20) (v2.8.1) with the parameter '-g 1600m' to indicate a genome size of 1600 Mb (representing the length of the diploid genome). The resulting assembly had a size of 1.33 Gb and a contig N50 of 440 kb (Table 2). Using this assembly, we performed two benchmarks: a first on the whole genome and a second on a specific genomic region which has been thoroughly analyzed in the Zhou *et al*. publication.

Metagenome assembly. We downloaded two MinION runs from a synthetic mock microbial community (23) composed of 12 bacterial strains (SRX4901586 and SRX5161985) as well as the corresponding Illumina sequencing reads (SRR8073716). We assembled the Nanopore data using Metaflye (24) and obtained a genome assembly of 43.1 Mb in size, composed of 104 contigs of >2 kb. Contigs were then polished two times with Racon (Medaka could not be launched on this too old dataset) and used as input assembly for all polishing methods.

Benchmarking of polishing methods

Each polisher was launched (on a 36 cores server with 380GB of memory) iteratively six times on the input assembly to evaluate accuracy and impact of multiple rounds of correction. If needed, Illumina reads were aligned with BWA mem (v0.7.17 with the default parameters except -t 36), and the resulting bam file was sorted and indexed using Samtools (17) (v1.10 with the default parameters except -@ 36 and -m 10G). Surprisingly, HyPo never succeeded when using a genome size of 1300 Mb for the correction of the *Solanum T.* genome assembly, but was able to polish the sequence when using 120 Mb. Importantly, as Apollo does not take into account paired-reads, short-reads alignment was performed in single-end mode. In our hands, Apollo needed 60 hours to perform a single iteration over the Arabidopsis genome, 60 times longer than most other methods. We therefore decided not to use it for further testing. The parameters used for each polisher are described in Table 1.

Polishing of homozygous genome assemblies

The QUAST suite (25) was used to generate statistics on the quality of the alignment between the polished assemblies and the reference genome (Col-0 downloaded from the TAIR website). In addition, Illumina reads were aligned on each polished assembly and only perfect alignments were kept. The accuracy and completeness of the gene content were assessed using TAIR10.1 annotation and BUSCO (26) (version 5 with brassicales_odb10 dataset). Exons from the reference *Arabidopsis* annotation (using the getfasta command from bedtools (27)) were extracted and aligned onto each assembly using the Blat aligner (28). Exons that were aligned with 100% identity along their entire length were kept and unique exon names were counted to avoid multi-mapping bias.

On the human genome dataset, all polishers were launched only one time, as this large assembly was mainly used to evaluate their restitution time and memory requirements. As with *Arabidopsis thaliana*, the quality was assessed by aligning the Illumina reads on each polished as-

Table 2. Datasets and long-read assemblies generated for the benchmark. Coverages were computed using a genome size of 120 Mb, 3 Gb, 1.6 Gb and 56 Mb for *Arabidopsis thaliana, Homo sapiens, Solanum tuberosum* and the metagenome sample (sum of genome sizes), respectively

		Arabidopsis thaliana Col-0	*Homo sapiens*	Synthetic sequence	*Solanum tuberosum* L. RH89–039-16	Metagenomic sample
Illumina	Accession number	SRR12136403	ERR194147	NA	PRJNA573826	SRX4901583
	Read length (bp)	2 × 150	2 × 101	2 × 150	2 × 250	2 × 151
	Coverage	176 X	30 X	50 X	47 X	1150 X
Nanopore	Accession number	SRR12136402	-	-	PRJNA573826	SRX5161985 SRX4901586
	Reads N50 (bp)	18 827	-	-	25 280	22 660
	Coverage	95 X	-	-	75 X	83 X
PACBIO HiFi	Accession number	-	-	-	SRX7922852	-
	Reads N50 (bp)	-	-	-	10 000	-
	Coverage	-	-	-	14 X	-
Assembly	Number of contigs	238	1,172	1	11,070	107
	Cumulative size	119 992 853	2 818 937 673	102 000	1 332 417 447	49 379 539
	Contig N50 (bp)	14 841 396	11 821 944	102 000	440 422	3 584 230

sembly and only perfect alignments were retained. The accuracy and completeness of the gene content was assessed with the reference annotation using pblat (29) and as previously described.

Polishing of heterozygous genome assemblies

The polished assemblies of the original 100 kb genomic region were aligned with the two haplotype sequences using muscle (30) (version 3.8), and each position was labeled haplotype 1 or haplotype 2 (if the base was similar to the corresponding haplotype), error (if the base was different from the two haplotypes) or equal (if the three bases were identical). For each polisher, the number of swaps between the two haplotypes and the number of errors were reported.

The diploid and heterozygous genome of *Solanum tuberosum* was used to assess the ability of each polisher to locally preserve the haplotype phasing. Two benchmarks were performed: a first on the whole genome and a second on a specific genomic region. Polishing algorithms were compared on their ability to phase genomic regions during the error-correction step. For that purpose, heterozygous variants were detected from the Illumina short reads, without prior assembly, using discoSNP (31) and default parameters. The sequence context (30 bp on the right and left side) of each variant was extracted from the discoSNP output and only 61 bp-sequence with a single variant were kept. In the discoSNP output, the detected variants were phased based on the Illumina reads (-A parameter of discoSNP), and only chains of at least three variants validated by at least 5 short reads were selected. For each heterozygous SNP, the two variants were mapped on each polished assembly and only perfect matches were kept. All reliable chains of variants were searched in the alignment results and a given chain was validated only if all its variants were found in a perfect match and on the same genomic sequence.

In addition, we focused on a 300 kb genomic region of chromosome 8, which has been described in the Figure 2B of the Zhou *et al*. publication. The authors illustrate a syntenic block on the two haplotypes. The coding exons of the two haplotypes were extracted and only the exons that contain at least one difference were selected and used as candidate exons. This region was assembled into four different contigs in our nanopore-based assembly, with two collapsed (contig_3372 and contig_15103) and two duplicated contigs (contig_15126 and contig_15127). The candidate exons were searched in the polished versions of these four contigs and only the perfect alignments were kept. The gene content completeness was evaluated using BUSCO (26) (version 5 with solanales_odb10 dataset).

Polishing of a metagenomic assembly

A synthetic mock community composed of 12 bacterial strains was used to compare each polisher and assess their ability to recover specific regions of each genome. For that purpose, we collected all 31-mers in the metagenome sample that are specific to a given species, using UniqueKMER (32) (Supplementary Table S1). We then searched for these specific 31-mers in each polished assembly and hypothesized that a higher number of specific 31-mers reflected a more realistic composition of the metagenome.

RESULTS

Impact of the sequencing coverage

Even if the sequencing coverage is no longer a problem, the *Arabidopsis thaliana* and *Solanum tuberosum* genome assemblies were polished independently with various depths of coverage. On the homozygous genome, the consensus quality is already high with as low as 25× of coverage, even if the overall quality can be improved with a higher coverage (between 50× and 100×, Supplementary Figure S1). Interestingly, some metrics (identity and number of perfect mapped reads) seem to decrease when the coverage is too high (180×, Supplementary Figure S1). In heterozygous regions, coverage has a direct impact on the ability to phase variants, and it intuitively requires twice as much coverage as in homozygous regions. With a coverage of 25×, Hapo-G missed 4% of phased variants, compared to polishing with 50×. However, the number of phased variants with 25× re-

mains similar or higher than that of other polishers, even with 50× of coverage (Supplementary Figure S2).

Polishing of homozygous genome assemblies

Overall, all of the polishing tools achieved very similar results in terms of quality metrics. They produced a polished assembly with <60 errors per 100 kb (Figure 3A), with the exception of ntEdit (75 errors per 100 kb), Apollo (96 errors per 100 kb) and Racon (128 errors per 100 kb). These results are confirmed by the number of perfectly mapped Illumina read pairs (Figure 3B), with the lowest scores obtained by Racon and ntEdit (34.1 and 41.8 M respectively), while the highest number was achieved by HyPo and NextPolish (43.0 M after the first round of correction). Regarding the alignment of the reference annotation, again, differences were small (<450 exons between Hapo-G, HyPo, NextPolish, Pilon and POLCA out of the 203 233 input exons), with ntEdit and Racon assemblies containing the lowest number of exons retrieved perfectly (Figure 3C). For Hapo-G, HyPo, ntEdit, NextPolish, Pilon and POLCA, increasing the number of polishing rounds didn't seem to have any significant impact, although two rounds for most tools seems to be optimal. Oddly, increasing the number of polishing rounds with Racon increased the number of errors per 100 kb from 128 for the first round to 161 for the sixth round. Racon's inferior performance can be explained by the fact that it was originally designed to perform polishing using long reads. Moreover, the gene content was very similar for all methods, and more complete than the uncorrected assembly, except for Racon and Apollo which obtained a lower value than the input assembly (Supplementary Table S2). The fastest average running time was achieved by ntEdit with approximately 25 min for each round, while the slowest was Apollo (3647 min for a single iteration) and Racon, with an average running time of 163 min. From the alignment-based methods, Hapo-G was the fastest with an average running time of 40 min (Figure 3D). Memory requirement is variable, ntEdit requires the least amount of memory, while alignment-based methods are generally limited by the use of samtools sort which is the part that requires the most amount of memory (Supplementary Figure S3). However, depending on the implementation, some methods require twice as much memory (Pilon, POLCA and Racon).

The human genome was successfully polished using Hapo-G, NextPolish, ntEdit and Pilon. Other methods were not able to produce results: HyPo crashed due to a lack of memory, Racon generated an empty fasta file with no error messages and POLCA crashed while processing the VCF (Variant Call Format) file. As already observed on the homozygous plant genome, the quality metrics were similar for all methods, and ntEdit had the lowest results but was the fastest. Hapo-G was the second fastest method, only 5 h compared to 30 h with NextPolish, and therefore seems to be a good compromise between speed and quality of the result (Supplementary Figure S4)

Polishing of heterozygous genome assemblies

Initially, the 100 kb sequence contained 861 haplotype switches and 1877 sequencing errors. In this benchmark, we only performed one round of correction for each tool. Pilon was the only polisher to generate more haplotype switches than there were initially in the reference (881 switches). The POLCA and ntEdit polished sequences contained >800 switches, while about 500 switches were still present in the HyPo, Racon and NextPolish corrected sequences. Hapo-G, the only tool dedicated to heterozygous genomes, obtained the best result with only 65 switches (Figure 4 and Supplementary Table S3). In terms of remaining sequencing errors, only three corrected sequences still contained some errors, the one obtained with: HyPo (5 errors), Racon (198 errors) and ntEdit (938 errors).

In the case of a more complex and heterozygous genome, the situation is different. Three methods seemed to perform better than the others: Hapo-G, HyPo and NextPolish. As for simple and homozygous genomes, ntEdit and Racon obtained the worst results (Figure 5A). However, the number of phased variants that could be recovered was higher in the assembly corrected with Hapo-G and this from the first round (472 534 phased variants compared to 469 441 after six rounds of NextPolish, Figure 5B). The situation is the same when looking at chains of at least three variants. Hapo-G was the only one to retrieve >100 000 chains whereas the second best result was obtained by NextPolish with 92 291 chains (Figure 5C). Additionally, Hapo-G was twice as fast as HyPo and seven times faster than NextPolish, the other two methods that performed well on heterozygous genomes. Furthermore, Hapo-G was the second fastest method and as previously observed, ntEdit was the fastest (Figure 5D). The six rounds of polishing using Hapo-G were faster than a single round performed with NextPolish. Interestingly, we have observed three types of tools: those that take advantage of multiple rounds of correction, those for which one round seems sufficient, and those that produce inferior results by performing multiple rounds of correction. Hapo-G, HyPo, Pilon and POLCA are in the first category, ntEdit and NextPolish in the second and Racon is the only one which seems to degrade the quality of the assembly as rounds are performed (Figure 5A–C).

In addition, we focused on the two allelic regions described in the study by Zhou *et al*. and counted the number of candidate exons in each assembly. In the Hapo-G corrected sequence, we recovered 86 candidate exons which represented the highest number of exons found in all assemblies, compared to 39 in the unpolished assembly. For comparison, 84 candidate exons were found in the NextPolish assembly which is the second best result (Supplementary Table S4).

Furthermore, the completeness of the gene content was evaluated on the whole genome assembly using BUSCO and 5950 genes conserved across solanales. Even if all the methods had a higher gene content completeness than the unpolished assembly (81.5%), only three methods (Hapo-G, HyPo and NextPolish) obtained a BUSCO score equal or higher than 98.2% (Supplementary Table S5).

Polishing of a metagenomic assembly

On the metagenomic sample, Hapo-G and HyPo were the two methods that allowed us to retrieve the highest number of specific 31-mers (Supplementary Figure S5B) and

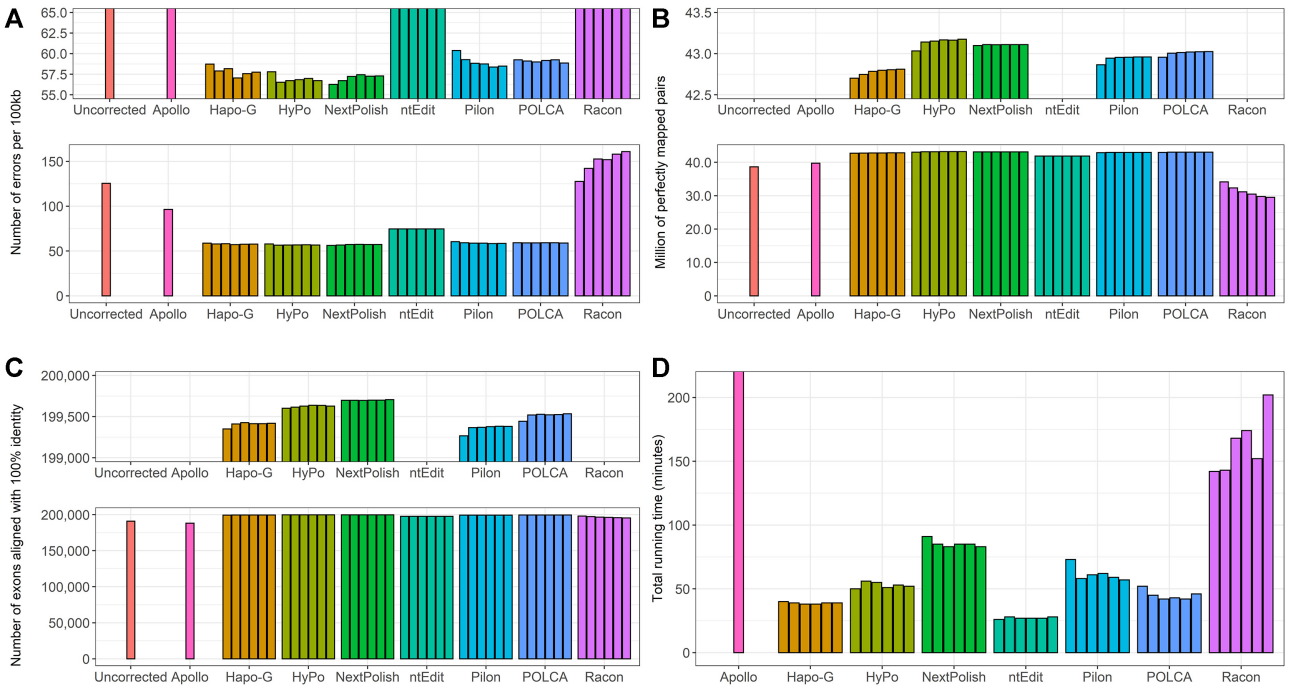

Figure 3. Comparison of polishing algorithms on the *Arabidopsis thaliana* genome assembly. Lower panels of A, B and C show the full distribution and the upper panels are a zoom on the higher values. (**A**) Number of errors per 100 kb after each round of polishing, when compared to the *Arabidopsis thaliana* reference genome. (**B**) Number of Illumina pairs mapped perfectly on each assembly. (**C**) Number of *Arabidopsis thaliana* exons aligned with 100% identity after each round of polishing. (**D**) Run times of polishing tools, for each polishing round.

Figure 4. Comparison of polishing algorithms on a synthetic diploid sequence. The 100 kb sequence is represented on the *x* axis and each polishing tool has a dedicated track, where remaining errors are represented with red bars and switches between the two haplotypes are represented by blue bars.

to perfectly align the higher number of paired-end reads (Supplementary Figure S5A). Hapo-G retrieved the highest number of specific 31-mers, and HyPo the highest number of paired-ends reads. This could be explained by the fact that the coverage was heterogeneous, so the number of specific 31-mers was not directly linked to the number of correctly mapped paired-ends reads. Interestingly, in this context, NextPolish is the fastest method (even faster than the kmer approach of ntEdit), only 30 mn of running time for the first iteration, compared to the ∼2 h needed for other mapping-based methods (Supplementary Figure S5C).

DISCUSSION

In this study, we report a new software, Hapo-G, which is able to polish draft assemblies with a quality equivalent to that of existing tools on simple and homozygous genomes, while being faster, but which also improves the polishing of

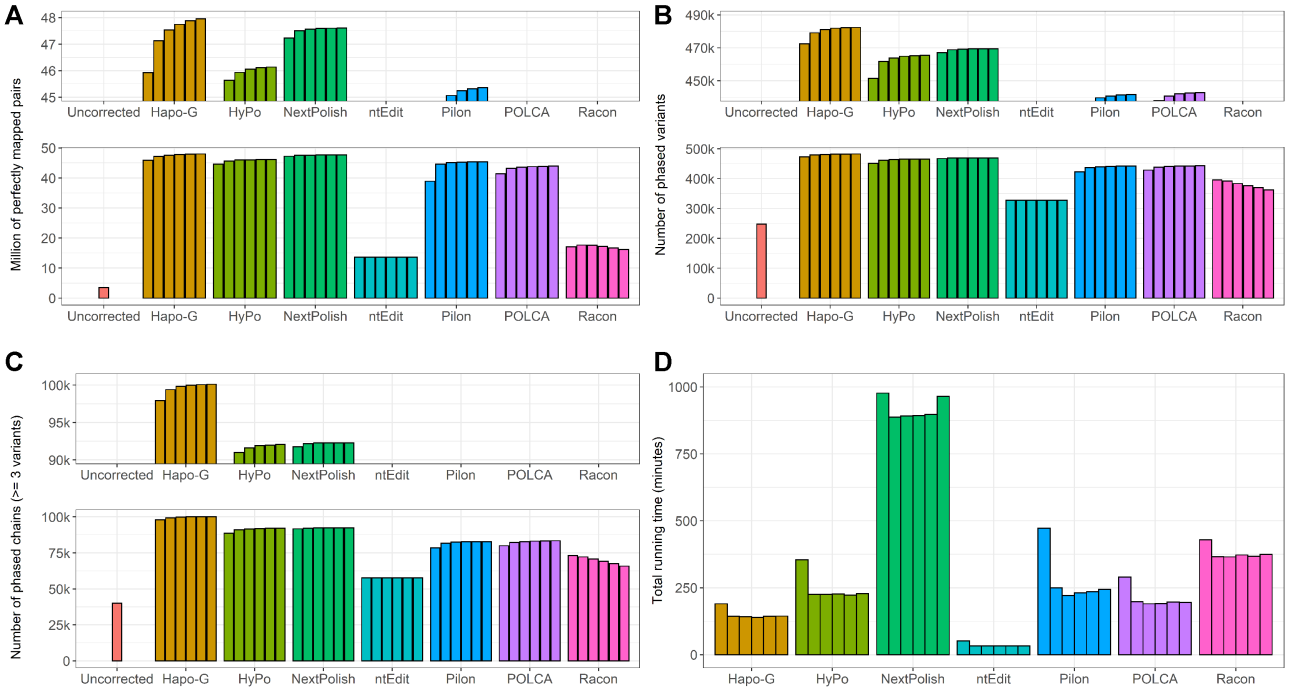

Figure 5. Comparison of polishing algorithms on the *Solanum tuberosum* genome assembly for each polishing round. Lower panels of A, B and C show the full distribution and the upper panels are a zoom on the higher values. (**A**) Number of Illumina pairs mapped perfectly on each assembly. (**B**) Number of phased variants retrieved in each assembly. (**C**) Number of chains composed of >3 variants (and confirmed by at least 5 reads) perfectly retrieved in each assembly. (**D**) Run time of each polishing tool.

heterozygous genomic regions as well as metagenomic assemblies.

Nowadays, the number of polishing tools is high and although almost all are based on the same principle (mapping of short reads), their performances are different, likewise none are specialized in processing heterozygous genomes. In this study, we compared eight existing algorithms: Hapo-G, HyPo, Apollo, NextPolish, ntEdit, Pilon, POLCA and Racon. We obtained very similar results on a small plant genome (*Arabidopsis thaliana*) with the exception of ntEdit which is the only tool not based on short-read alignment. In addition, we observe that the oldest and most widely used polishing tool, Pilon, is not among the tools with the best results. In general, Hapo-G, HyPo and NextPolish stand out and often give good results.

We observed on a synthetic diploid sequence and on a true heterozygous genome (*Solanum tuberosum*) that the phasing of variants is of better quality when the assembly is polished with Hapo-G, leading after six rounds of correction to the higher number of paired-end reads perfectly mapped back to the assembly (47 957 836 out of 151 018 344). Only two tools, ntEdit and Hapo-G, succeeded in polishing the 1.3 Gb of the *Solanum tuberosum* genome assembly in <3 h on average. On this large genome, the six rounds of Hapo-G ended before the first round of NextPolish, which is the second best tool according to our benchmark and already incorporates several rounds of mapping/correction internally. Importantly, on much larger genomes like that of the Human, HyPo, Apollo and Racon failed to polish the consensus in <3 days on a 36-core server with 380 Gb of memory.

Increasing the number of correction rounds is generally beneficial, except for ntEdit and NextPolish where the results are very similar from round one through sixth, and Racon where the quality of the consensus seems to deteriorate when adding new correction cycles.

In homozygous regions, although many polishers have achieved similar results, NextPolish appears to be the best performer, therefore, if possible, we suggest using NextPolish in combination with Hapo-G to achieve high quality in homozygous and heterozygous regions. It is important to note that Hapo-G should be used last to best preserve variant phasing. In fact, homozygosity is generally not complete and heterozygous regions may remain. Interestingly, by combining NextPolish and Hapo-G, the number of perfectly mapped paired-end reads was higher than after six rounds of Hapo-G or NextPolish separately. However, a combination of Hapo-G with any other polisher did not lead to better results in our tests (Supplementary Figure S6).

Hapo-G algorithm requires high-quality reads, which excludes error-prone long reads. However, PACBIO HiFi long-reads are suitable to polish a genome assembly using Hapo-G. Even if it was not initially dedicated to long reads, we polished the *Solanum tuberosum* using both short and long reads. The input coverage was low (14×), but combining both technologies allowed Hapo-G to increase the number of phased variants (491 093 compared to 482 404 after six rounds of Hapo-G with short reads, Supplementary Figure S7).

Hapo-G is often the fastest alignment-based method, achieves good results (particularly in metagenomic samples and heterozygous regions) and is flexible (it can take long or short reads as input as long as they are of high quality). Based on these observations, we recommend using Hapo-G to polish genome and metagenome assemblies and eventually by performing multiple rounds of correction if possi-

ble. In addition, depending on the size of the genome and if possible, we recommend combining Hapo-G with another polisher like HyPo or NextPolish which generally give good results.

DATA AVAILABILITY

Hapo-G is an open source software, source code, binaries as well as results of the benchmark are freely available from http://www.genoscope.cns.fr/hapog. In addition, Hapo-G has been added in the Anaconda repository for ease of installation and use: https://anaconda.org/lbgb_cea/hapog. All data, short and long reads, used in the article are available on public repositories.

SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

ACKNOWLEDGEMENTS

The authors thank Pierre Peterlongo for his support and advice with discoSNP and his proofreading of the manuscript.

FUNDING

Agence Nationale de la Recherche [ANR-10-INBS-09–08].
Conflict of interest statement. None declared.

REFERENCES

1. Belser,C., Istace,B., Denis,E., Dubarry,M., Baurens,F.-C., Falentin,C., Genete,M., Berrabah,W., Chèvre,A.-M., Delourme,R. *et al.* (2018) Chromosome-scale assemblies of plant genomes using nanopore long reads and optical maps. *Nat. Plant.*, **4**, 879–887.
2. Jain,M., Koren,S., Miga,K.H., Quick,J., Rand,A.C., Sasani,T.A., Tyson,J.R., Beggs,A.D., Dilthey,A.T., Fiddes,I.T. *et al.* (2018) Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat. Biotechnol.*, **36**, 338–345.
3. Schmidt,M.H.-W., Vogel,A., Denton,A.K., Istace,B., Wormit,A., van de Geest,H., Bolger,M.E., Alseekh,S., Maß,J., Pfaff,C. *et al.* (2017) De Novo Assembly of a New Solanum pennelliiAccession Using Nanopore Sequencing. *Plant Cell*, **29**, 2336.
4. Liu,J., Seetharam,A.S., Chougule,K., Ou,S., Swentowsky,K.W., Gent,J.I., Llaca,V., Woodhouse,M.R., Manchanda,N., Presting,G.G. *et al.* (2020) Gapless assembly of maize chromosomes using long-read technologies. *Genome Biol.*, **21**, 121.
5. Rousseau-Gueutin,M., Belser,C., Da Silva,C., Richard,G., Istace,B., Cruaud,C., Falentin,C., Boideau,F., Boutte,J., Delourme,R. *et al.* (2020) Long-read assembly of the Brassica napus reference genome Darmor-bzh. *GigaScience*, **9**, giaa137.
6. Alonge,M., Wang,X., Benoit,M., Soyk,S., Pereira,L., Zhang,L., Suresh,H., Ramakrishnan,S., Maumus,F., Ciren,D. *et al.* (2020) Major Impacts of Widespread Structural Variation on Gene Expression and Crop Improvement in Tomato. *Cell*, **182**, 145–161.
7. Jiao,W.-B. and Schneeberger,K. (2020) Chromosome-level assemblies of multiple Arabidopsis genomes reveal hotspots of rearrangements with altered evolutionary dynamics. *Nat. Commun.*, **11**, 989.
8. Song,J.-M., Guan,Z., Hu,J., Guo,C., Yang,Z., Wang,S., Liu,D., Wang,B., Lu,S., Zhou,R. *et al.* (2020) Eight high-quality genomes reveal pan-genome architecture and ecotype differentiation of Brassica napus. *Nat. Plants*, **6**, 34–45.
9. Walker,B.J., Abeel,T., Shea,T., Priest,M., Abouelliel,A., Sakthikumar,S., Cuomo,C.A., Zeng,Q., Wortman,J., Young,S.K. *et al.* (2014) Pilon: An integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One*, **9**, e112963.
10. Watson,M. and Warr,A. (2019) Errors in long-read assemblies can critically affect protein prediction. *Nat. Biotechnol.*, **37**, 124–126.
11. Vaser,R., Sović,I., Nagarajan,N. and Šikić,M. (2017) Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Res.*, **27**, 737–746.
12. Firtina,C., Kim,J.S., Alser,M., Senol Cali,D., Cicek,A.E., Alkan,C. and Mutlu,O. (2020) Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm. *Bioinformatics*, **36**, 3669–3679.
13. Warren,R.L., Coombe,L., Mohamadi,H., Zhang,J., Jaquish,B., Isabel,N., Jones,S.J.M., Bousquet,J., Bohlmann,J. and Birol,I. (2019) ntEdit: scalable genome sequence polishing. *Bioinformatics*, **35**, 4430–4432.
14. Hu,J., Fan,J., Sun,Z. and Liu,S. (2020) NextPolish: a fast and efficient genome polishing tool for long-read assembly. *Bioinformatics*, **36**, 2253–2255.
15. Kundu,R., Casey,J. and Sung,W.-K. (2019) HyPo: super fast & accurate polisher for long read genome assemblies. bioRxiv doi: https://doi.org/10.1101/2019.12.19.882506, 20 December 2019, preprint: not peer reviewed.
16. Zimin,A.V. and Salzberg,S.L. (2020) The genome polishing tool POLCA makes fast and accurate corrections in genome assemblies. *PLOS Comput. Biol.*, **16**, e1007981.
17. Li,H., Handsaker,B., Wysoker,A., Fennell,T., Ruan,J., Homer,N., Marth,G., Abecasis,G., Durbin,R. and 1000 Genome Project Data Processing Subgroup (2009) The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.
18. Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, **25**, 1754–1760.
19. Li,H. (2018) Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, **34**, 3094–3100.
20. Kolmogorov,M., Yuan,J., Lin,Y. and Pevzner,P.A. (2019) Assembly of long, error-prone reads using repeat graphs. *Nat. Biotechnol.*, **37**, 540–546.
21. Huang,W., Li,L., Myers,J.R. and Marth,G.T. (2012) ART: a next-generation sequencing read simulator. *Bioinformatics*, **28**, 593–594.
22. Zhou,Q., Tang,D., Huang,W., Yang,Z., Zhang,Y., Hamilton,J.P., Visser,R.G.F., Bachem,C.W.B., Robin Buell,C., Zhang,Z. *et al.* (2020) Haplotype-resolved genome analyses of a heterozygous diploid potato. *Nat. Genet.*, **52**, 1018–1023.
23. Sevim,V., Lee,J., Egan,R., Clum,A., Hundley,H., Lee,J., Everroad,R.C., Detweiler,A.M., Bebout,B.M., Pett-Ridge,J. *et al.* (2019) Shotgun metagenome data of a defined mock community using Oxford Nanopore, PacBio and Illumina technologies. *Sci. Data*, **6**, 285.
24. Kolmogorov,M., Bickhart,D.M., Behsaz,B., Gurevich,A., Rayko,M., Shin,S.B., Kuhn,K., Yuan,J., Polevikov,E., Smith,T.P.L. *et al.* (2020) metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nat. Methods*, **17**, 1103–1110.
25. Gurevich,A., Saveliev,V., Vyahhi,N. and Tesler,G. (2013) QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, **29**, 1072–1075.
26. Waterhouse,R.M., Seppey,M., Simão,F.A., Manni,M., Ioannidis,P., Klioutchnikov,G., Kriventseva,E.V. and Zdobnov,E.M. (2018) BUSCO applications from quality assessments to gene prediction and phylogenomics. *Mol. Biol. Evol.*, **35**, 543–548.
27. Quinlan,A.R. and Hall,I.M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841–842.
28. Kent,W.J. (2002) BLAT—The BLAST-Like Alignment Tool. *Genome Res.*, **12**, 656–664.
29. Wang,M. and Kong,L. (2019) pblat: a multithread blat algorithm speeding up aligning sequences to genomes. *BMC Bioinformatics*, **20**, 28.
30. Edgar,R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.
31. Peterlongo,P., Riou,C., Drezen,E. and Lemaitre,C. (2017) DiscoSnp++: de novo detection of small variants from raw unassembled read set(s). bioRxiv doi: https://doi.org/10.1101/209965, 27 October 2017, preprint: not peer reviewed.
32. Chen,S., He,C., Li,Y., Li,Z. and Melançon,C.E. III (2021) A computational toolset for rapid identification of SARS-CoV-2, other viruses and microorganisms from sequencing data. *Brief. Bioinform.*, **22**, 924–935.