BIOINFORMATICS ORIGINAL PAPER

Vol. 29 no. 3 2013, pages 316–321
doi:10.1093/bioinformatics/bts712

Genome analysis

Advance Access publication December 24, 2012

SRmapper: a fast and sensitive genome-hashing alignment tool

Paul M. Gontarz, Jennifer Berger and Chung F. Wong*

Department of Chemistry and Biochemistry, Center for Nanoscience, University of Missouri-Saint Louis, One University Boulevard, St. Louis, MO 63121, USA

Associate Editor: Martin Bishop

ABSTRACT

Summary: Modern sequencing instruments have the capability to produce millions of short reads every day. The large number of reads produced in conjunction with variations between reads and reference genomic sequences caused both by legitimate differences, such as single-nucleotide polymorphisms and insertions/deletions (indels), and by sequencer errors make alignment a difficult and computationally expensive task, and many reads cannot be aligned. Here, we introduce a new alignment tool, SRmapper, which in tests using real data can align 10s of billions of base pairs from short reads to the human genome per computer processor day. SRmapper tolerates a higher number of mismatches than current programs based on Burrows–Wheeler transform and finds about the same number of alignments in 2–8× less time depending on read length (with higher performance gain for longer read length). The current version of SRmapper aligns both single and pair-end reads in base space fastq format and outputs alignments in Sequence Alignment/Map format. SRmapper uses a probabilistic approach to set a default number of mismatches allowed and determines alignment quality. SRmapper's memory footprint (~2.5 GB) is small enough that it can be run on a computer with 4 GB of random access memory for a genome the size of a human. Finally, SRmapper is designed so that its function can be extended to finding small indels as well as long deletions and chromosomal translocations in future versions.

Availability: http://www.umsl.edu/~wongch/software.html.

Contact: wongch@umsl.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on September 5, 2012; revised on November 16, 2012; accepted on December 16, 2012

1 INTRODUCTION

With the advent of next-generation sequencing (NGS) instruments, the amount of raw genetic sequence information has exponentially increased during the past few years, and it is expected to continue to grow at a high rate as sequencing cost continue to decrease. Instruments such as the HiSeq2000 (Illumina), GS FLX titanium (Roche) and SOLiD 4 (ABI) can generate billions of base pairs (gigabases or Gb) of data per day with increasingly high accuracies, >98.5% for Illumina and >99.5% for Roche and ABI, respectively, and with costs that have decreased to ~$10 000 per human genome (Pareek *et al*., 2011). Newer instruments, such as the Ion Proton (Life Technologies), can produce even higher amounts of data and are approaching the goal of a $1000 genome (Rothberg *et al*., 2011). With the speed and cost factors making whole-genome sequencing practical, researchers are sequencing and analysing large numbers of genomes in hopes of finding genetic origins of many diseases, such as cancers. For example, one recent study sequenced 457 human genomes searching for rare mutations involved in ovarian cancer (Rafnar *et al*., 2011). With the dramatically increased amount of raw data, analysis has become more challenging. This is because of two factors: the sheer amount of data gathered and the relatively short lengths of reads produced by current NGS instruments (Ng and Kirkness, 2010). For example, in the study aforementioned, >12 terabases of sequence data would be gathered for a 10× coverage of the 457 genomes. The short length of the reads, typically between 30 and 100 bp for Illumina and ABI and ~400 bp for Roche, complicates the building of a genome *de novo* (Ng and Kirkness, 2010; Pareek *et al*., 2011). For example, a genome may contain repetitive regions. It is difficult to reconstruct these regions and their flanking sequences if the length of the reads is much shorter than that of the repeating units. NGS instruments can now perform pair-end sequencing, in which the sequence of the two ends of longer fragments are determined, which has helped to resolve these problems (Ng and Kirkness, 2010; Pareek *et al*., 2011). Nevertheless, *de novo* assembly remains challenging and is memory intensive and, therefore, difficult to perform on genomes larger than those of bacteria (Pareek *et al*., 2011).

A popular alternative to *de novo* assembly is reference assembly, in which reads are aligned against a pre-existing reference genome. There are three main classes of alignment tools: read-hashing tools, reference-hashing tools and Burrows–Wheeler transform (BWT) tools. Examples of reference-hashing tools include BFAST (Homer *et al*., 2009), SHRiMP-2 (David *et al*., 2011) and WHAM (Li *et al*., 2011). Tools that use the BWT include Burrows–Wheeler Aligner (BWA) (Li and Durbin, 2009, 2010), bowtie (Langmead *et al*., 2009) and SOAP2 (Li *et al*., 2009b). Most genome-hashing algorithms require large memory that they must be run on expensive large-memory machines. On the other hand, many BWT methods carry a small memory footprint that they can be run on computers with 4 GB of random access memory (RAM), accessible by many users possessing only desktop machines. Among the genome-hashing methods listed previously, only BFAST can be run on a computer with 4 GB of RAM. For methods based on BWT, both BWA and bowtie can be run on computers with 4 GB of RAM. Among these three, bowtie is the most restrictive in terms of allowing variants between the reference and short read, allowing a maximum of three mismatches and no insertions or deletions

*To whom correspondence should be addressed.

(indels), making it a less attractive option for sequencing longer reads, which would be expected to have a higher number of mismatches and errors. BWA and BFAST both allow indels and mismatches. Among the three, bowtie is slightly faster than BWA, and both are significantly faster than BFAST. However, BFAST has been shown to be more sensitive than the BWT-based methods for most datasets evaluated (Table 2 in Homer *et al*., 2009).

Here, we introduce a new genome-hashing alignment tool, SRmapper. The original design goal for SRmapper was to build an alignment tool that was not restrictively slow, had as high sensitivity as other widely used alignment tools and was capable of finding long deletions and other more complicated genomic alternations, such as chromosomal translocations from short-read sequences. The current version of SRmapper already achieves the first two goals and uses a novel approach to determine the initial number of mismatches allowed and calculates alignment scores. In evaluations on real data, SRmapper is 2–8× faster than BWA on datasets of length ≥51 bp while aligning comparable number of reads as BWA. For short reads, 32 bp, SRmapper was 6–40× faster than BWA, but somewhat less sensitive. This article explains the approach taken by SRmapper, compares its performance against the popular BWA package for multiple datasets of different read length, describes how we envision this first version of SRmapper being used and discusses future extensions and improvements.

2 METHODS

2.1 Indexing reference genomes

SRmapper takes reference data in the form of fasta files. It first builds an index from the reference genome or sequences to which short reads are to be aligned and writes the index to file, such that the index only has to be built once. The index takes the form of a hash table. Hashing is done by first transforming a hash key containing a sequence of D bases into a base-4 number with D digits. Each digit represents a base in the sequence by taking a value of 0(A), 1(C), 2(G) or 3(T). This base-4 number is then converted into a base-10 value, V10, which points to a bucket containing the locations of the genome in which this sequence is found. To reduce the size of the index, SRmapper divides a genome into non-overlapping sequences containing D bases and stores the locations of these sequences in the corresponding buckets using the hash function previously described. Non-standard nucleotides, N, can be converted to random bases or keys containing a non-standard nucleotide can be skipped; the current version takes the latter approach. For a reference sequence of length R, D is chosen according to $D(R) = \text{floor}\,[\log_4(R)]$. This value is selected in an attempt that for any R, a hash table with minimal collisions will be constructed while also minimizing memory usage. For the human genome with ~3 billion bases, D = 15; for a virus with a 10-kb genome, D = 6. The number of buckets is 4^D. The total number of locations stored in the buckets is ≈R/D.

For D = 15, all buckets cannot be stored in 4 GB of memory at the same time because each bucket requires ~4 bytes to hold the largest number from the reference sequence of the human genome (4 bytes/bucket × 4^{15} buckets = 4 GB). We avoided this problem by storing only 4D-1 buckets at a time and making four passes through the reference sequence. In the first pass, all the locations in the index that start with 'A' are loaded into memory; in the second, all locations in the index starting with 'C' are loaded into memory, and so on. In practice, the buckets are not declared until a key corresponding to the particular bucket is found in the genome. This reduces memory usage, as all keys are not found in the genome. At the end of each pass, the locations in the buckets alone are written to a binary file. A count is kept of how many locations have been dumped, and after each key's bucket has been written, the current value of the counter is written to a separate binary file (Supplementary Fig. S1 illustrates this with a simple example). SRmapper also builds a compressed, 2 bit per base form of the reference genome or sequences. Finally, the indexing program builds a file containing header information from the fasta files, including chromosomal lengths and chromosome numbers to be used in output downstream.

2.2 Short-read alignment

Reads and read length: SRmapper takes short-read data in the form of one or more fastq files. In the current version, reads must be in base space. There is no hard limit on read length, but SRmapper's sets a default maximum read length of 1000 bp. Users aligning reads >1000 bp can define their own maximum read length at run time. SRmapper does not require all reads to be of the same length. This allows users to pre-trim reads to a specific length or to trim low-quality bases off the end of reads without causing SRmapper to fail. SRmapper does require reads to be of a minimum length of D.

Setting initial mismatches allowed: SRmapper was built to be flexible with respect to both the reference being aligned against and read length. SRmapper considers both read length and reference genome length when determining how many mismatches should be allowed for a particular read. Before aligning reads, SRmapper builds a mismatch table of maximum mismatches allowed for every read length up to maximum allowed read length and calculates a mapping quality associated with the number of mismatches in a read. This allows a lookup in the table to determine mismatches allowed instead of performing calculations to determine mismatches allowed for every read. Alternatively, users are allowed to set the maximum number of mismatches between a read and reference.

Calculating initial mismatches allowed and phred score: To simplify calculations on determining mismatches allowed, two assumptions are made about reference sequences. First, it is assumed that for all j and k with $j \neq k$, the identity of nucleotide j (N_j) and nucleotide k is independent of each other. Second, we assume that for any j, there is an equal probability that N_j is A, C, G or T. A probability function for whether an alignment had been generated spuriously or not is then generated as follows. The probability that a base will match at a random location in the genome is 1/4. For a read of length L, there are 4^L possible combinations, and thus a $1/4^L$ probability of a perfect match at a specific location. If a read was allowed to have one mismatch in it, there would be L possible locations for the mismatch, and each location would have three possible changes to a base that would cause a mismatch. Thus, the probability that a read of length L aligns to a specific location with one mismatch is $3L/4^L$ or $3^1{}_LC_1/4^L$. For two mismatches, there are L locations possible for the first mismatch and L-1 locations possible for the second mismatch, and as the order of mismatches is not important, we divide L (L-1) by two. Again, for each location, there are three possible changes to a base that result in a mismatch. Thus, the probability of an alignment with two mismatches is $3^2{}_LC_2/4^L$. It holds for any number of mismatches, M, that the probability of an alignment at a random location is $(3^M{}_LC_M)/4^L$ and that the probability of an alignment with M or fewer mismatches is

$$P(L, M) = \left(\sum_{i=0}^{M} 3^i{}_LC_i\right)/4^L \quad (1)$$

The probability that an alignment is not generated at a random location is then $1 - P(L, M)$. The probability of a match not occurring at any location within the genome is the cumulative probability of a match not occurring at each possible location in a genome of length R:

$$P(L, M, R) = (1 - P(L, M))^R = \left(1 - \left(\sum_{i=0}^{M} 3^i{}_LC_i\right)/4^L\right)^R \quad (2)$$

The probability that a read does align to the genome is then $1 - P(L, M, R)$. Phred score (Ewing and Green, 1998; Ewing *et al.*, 1998), which was originally used to describe experimental error, is used here to estimate the probability of aligning a read with a certain number of mismatches to a reference genome by chance, so as to aid SRmapper to decide how many mismatches to consider in aligning read with a certain length. Phred score is a logarithmically scaled number that measures the probability that a base is incorrect and is scaled up by a factor of 10. We estimate alignment qualities by using a phred-like value that uses the probability of Equation 2:

$$phred(L, M, R) = -10\log\left(1 - \left(1 - \left(\sum_{i=0}^{M} 3^i{}_L C_i\right)/4^L\right)^R\right) \quad (3)$$

Equation 3 can be approximated well by

$$phred(L, M, R) = -10\log\left(R\left(\sum_{i=0}^{M} 3^i{}_L C_i\right)/4^L\right) \quad (4)$$

Supplementary Figure S2 shows an example that Equation 4 approximates Equation 3 readily. By default, the maximum number of mismatches, M_m, allowed for a read is set, such that an alignment has a phred of 30 (>99.9% chance of not being randomly aligned); however, the user can specify any minimum phred score for alignment and SRmapper will cap the allowed number of mismatches, such that all alignments will have the minimum allowed score. Reads having multiple alignments with the same number of mismatches are automatically assigned a phred score of zero and an alignment location placed randomly from among the reads. Random placement from among equally scoring hits is used by other alignment tools as well.

Alignment of short reads to reference genomes: SRmapper takes as input base space reads in fastq format. Alignment is performed in two steps as the seed-and-extend strategy taken by the short-read aligners SSAHA (Ning *et al.*, 2001) and Stampy (Lunter and Goodson, 2011), but using different methods of seeding and extension. In the first step, the first D bases (1 through D) from the read are passed through the same hash function as in indexing. SRmapper then takes values from the corresponding bucket from the hashing to create initial possible alignments. In the second step, the remaining bases are compared with their corresponding bases in the reference as determined by the first step (Supplementary Fig. S3 illustrates with a simple example). Comparison is terminated either when all bases in the read have been checked against the reference or when the number of mismatches between the read and reference exceeds the maximum allowed number of mismatches, M_m. A possible alignment is considered a proper alignment if, on comparing all bases against the reference, the number of mismatches is less than or equal to the allowed number of mismatches. If an alignment is found, M_m is then decreased to the number of mismatches in that alignment, and the location of the alignment and the number of mismatches is stored. The first and second steps are repeated using bases 2 through D+1, and this process is repeated until all L bases in the read have been used in the first step of alignment or until D ($M_m + 2$) bases have been used in the first step of alignment. The latter case applying to longer reads in which the whole read does not need to pass through step 1 is justified as follows: If there is no mismatch, using the first 2D bases to create the fragments 1 to D, 2 to D + 1, . . . D+1 to 2D as seed to search the index, containing only non-overlapping segments of the genome, will guarantee an exact match to be found, if it exists. If there is one mismatch, the worst case scenario is to have a mismatch occurring at the Dth base of a read, so that a continuous stretch of D bases occurring in the index could only occur afterwards. Again, because we only store the genome in non-overlapping, rather than overlapping, segments of D bases to keep the size of the index small, we have to search 2D, rather than D, bases to ensure a contiguous segment of D bases can be found if the read can be mapped to the genome. Thus, the first $D \times 1 + 2D$ bases in a read need to be used. The general expression for any number of mismatch k is $D \times k + 2\ D = D \times (k + 2)$. After checking for alignments on the forward strand, the reverse complement of the read is generated to search for alignments on the reverse strand. The value of M_m is not reset to its original value when performing alignment on the reverse strand but remains at its current value as found during alignment on the forward strand.

Output format and options: Similarly to indexing, SRmapper performs alignment in four passes with a quarter of the index (corresponding to the A, C, G or T buckets) being stored in memory at any one time. The locations of the best hits from each quarter of the index are stored in corresponding binary files, which are deleted after a permanent alignment file is built. The final output is presented in Sequence Alignment/Map (SAM) format (Li *et al.*, 2009a). For single-end reads, the user may specify a maximum number of equally best alignments to output. By default, this value is set at one. For pair-end reads, only the best alignment will be displayed. SRmapper does not report reads failing to be aligned in its SAM file. Instead, it writes these reads to a separate file in a minimal fastq format, so that other alignment tools can attempt to align reads that SRmapper was unable to align.

Pair-end alignment: This proceeds similarly to single-end alignment until after the binary temp files are written. Possible alignments are generated treating each mate in a pair-end read as single-end reads, and after the alignments generated for both mates in the pairs, pair-end reads are searched for. Two single-end alignments are considered to form a proper pair-end alignment if they are mapped to within a sufficiently close distance. By default, this distance is set to 1 kb, but a value more appropriate for a specific dataset being analysed can be chosen by the user at run time. In single-end alignment, four binary temp files are written, whereas in pair-end alignment, eight are written. For the two binary files for each quarter of the index aligned against, one comes from alignments on the first reads in the pairs, whereas the second comes from the second mates. Unlike single-end alignment in which only the reads with the highest phred are selected, pair-end alignments also consider lower quality alignments that were found during the course of the single-end alignment. If multiple possible pair-end alignments are found, the pair-end alignment with the fewest combined mismatches is chosen, and the reported phred score is the score for a single-end read with the same length and number of mismatches as the length and mismatches of the two mates combined. If multiple reads of equal quality are found, one is chosen, at random, and the reported phred score is zero.

Increasing alignment speed: Apart from the D (k + 2) base search for k mismatch reads, SRmapper imposes two other limits by which alignment speed is substantially increased. The first is to decrement the number of allowed mismatches by one if a set number of alignments with an equal number of mismatches are found for a read. By default, this value is set at five for each quarter of the index, but this value can also be changed by the user at run time. This modification has no effect on the number of confident alignments achieved (those with a phred score greater than zero), as only reads with multiple, equally best alignments are affected. The second is to limit how many locations will be looked at from each bucket. This trade's speed for the number of reads aligned, as not looking in at locations will cause some reads to be missed but will require less time. This policy is enforced because of the large number of locations found in buckets corresponding to low complexity regions, such as poly-A tracts, dGdC islands, short interspersed elements (SINES) or long interspersed elements (LINES). By default, SRmapper only considers a maximum of 100 locations per bucket. This limit affects under 1:26 000 buckets and results in >99.5% of reads without this imposed limit still being aligned while reducing alignment time by >9×. The number of locations checked per bucket can be modified, or limited checking of the buckets can be disabled by the user at run time.

3 RESULTS

3.1 Indexing reference genomes

For both SRmapper and BWA, indexing the genome only needs to be performed once, as their respective indices are written to file after being built. SRmapper takes as input one or more chromosomal fasta files, whereas BWA requires these files to first be concatenated into one fasta file to index the whole genome or a pre-concatenated file with all chromosomes to be used. Indexing the human genome with SRmapper required 2350 s on an Intel Xeon 2.8 GHz processor. BWA required 8100 s to perform the same task, making SRmapper ~3.5× faster in the indexing stage. SRmapper does require somewhat more disk space for its index, as it uses 5.4 GB in comparison with BWA's 4.3 GB.

3.2 Evaluation of alignment performance on real data

To measure the performance of SRmapper against BWA, we downloaded four sets of data from the sequence read archive (http://www.ncbi.nlm.nih.gov/sra): SRR002787, which consists of 5.88 M 32-bp single-end reads, SRR006150, which consists of 13.18-M pair-end reads with each mate in the pair being 51 bp, SRR020477, which consists of 2.04-M pair-end reads with each mate being 76 bp, and SRR539393, which contains 2.25-M pair-end reads sequenced by the Illumina HiSeq 2000 from individual NA12878. Additionally, SRR006150, SRR020477 and SRR539393 were also evaluated as single-end reads. For SRR006150, the first mate was taken to create 13.18 M 51-bp single-end reads. For SRR020477 and SRR539393, both mates in the pair were evaluated to create 4.08 M 76-bp and 4.50 M 100-bp single-end reads, respectively. Reads were downloaded in Short Reads Archive (SRA) format and converted to fastq format by the SRA Toolkit's fastq-dump command (the SRA Toolkit is a resource provided by the National Center for Biotechnology Information). We used BWA 0.5.8c for all comparisons against BWA. BWA accomplishes alignment over two steps. In the first, the suffix array coordinates are determined for short reads, and in the second, the suffix array values are converted into chromosomal coordinates and output in SAM format. Thus, the time for BWA alignment is the sum of the time for the two aforementioned steps. As this first version of SRmapper does not perform gapped alignment, we disabled gapped alignment on BWA. Tests were run both using the default number of mismatches allowed by BWA and by SRmapper.

For single-end read alignment, SRmapper was several times faster with results varying from 2.1 to 8.7× faster (Table 1). The sensitivity of SRmapper, in terms of the percentage of reads that could be aligned, was comparable with BWA or better except for the shortest reads of 32 bp. These results are not surprising because of the different methods by which the tools perform alignment. As SRmapper needs to find an exact match for D bases in its initial placement of a read, some reads with high mismatch rates will not be aligned because SRmapper cannot find a D base stretch with no mismatches when it performs its initial alignment. This is more problematic with shorter reads, as there are fewer initial alignments (L–D + 1) to use. On the other hand, BWA allows mismatches in its seed sequence (with computational costs that increase with the number of mismatches allowed in the seed). Nevertheless, in all tests, there was a high overlap of reads found by both tools qualitatively, suggesting that as BWA has been shown to be accurate (Li and Durbin, 2009) and SRmapper produces similar results to BWA, SRmapper also accurately maps reads. In all cases with significantly different sensitivity, the more sensitive tool found >99% of reads found by the less sensitive tool while aligning additional reads missed by the less sensitive tool because of the reasons previously listed. In pair-end evaluations, SRmapper retained its speed advantage over BWA, with speeds being between 2–8× faster depending on the dataset and parameters used. Pair-end evaluations on BWA were run with the –A option to disable Smith–Waterman alignment. Enabling Smith–Waterman alignment increases the amount of reads mapped by BWA but also increases the alignment time. For pair-end alignment, BWA has an improved sensitivity compared with single-end read results (Table 2).

Table 1. Comparison of the performance of SRmapper with BWA on single-end reads

	Alignment time (s)	Speed-up	% aligned
SRR002787			
BWA default	4404		56.84
BWA @ SRmapper def	29 023		66.08
SRmapper @ BWA def	715	6.16	50.74
SRmapper default	744	39.01	56.62
SRR006150			
BWA default	10 673		73.49
BWA @ SRmapper def	13 574		82.18
SRmapper @ BWA def	2771	3.85	72.68
SRmapper default	3611	3.76	84.08
SRR020477			
BWA default	1482		66.35
BWA @ SRmapper def	8112		78.35
SRmapper @ BWA def	616	2.41	67.62
SRmapper default	930	8.72	84.02
SRR539393			
BWA default	2715		90.0
BWA @ SRmapper def	11 155		95.6
SRmapper @ BWA def	1298	2.09	90.2
SRmapper default	1548	7.21	91.0

Alignment time for 5.88 million 32-bp reads (SRR002787), 26.28 million 51-bp reads (SRR006150), 4.08 million 76-bp reads (SRR020477) and 4.49 million 100-bp reads (SRR5393939) downloaded from the sequence read archive (http://www.ncbi.nlm.nih.gov/sra) to the human genome was determined. BWA was run with the –o 0 option to disable gapped alignment. For BWA@def, BWA was run with its default mismatch settings. For BWA@SRmapper, BWA was run allowing the default number of mismatches permitted by SRmapper. SRmapper was run with its default settings in SRmapper@def and using the default number of mismatches permitted by BWA for SRmapper@BWA. Alignment time for BWA is the sum of the time for commands aln and sampe to run. Default mismatches allowed for SRR002787 are 2 and 3 for BWA and SRmapper, respectively, for SRR006150 are 3 and 12 for BWA and SRmapper, respectively, for SRR020477 are 4 and 28 for BWA and SRmapper, respectively, and for SRR539393 are 5 and 40 for BWA and SRmapper, respectively.

Table 2. Comparison of the performance of SRmapper with BWA on pair-end reads

	Alignment time (s)	Speed-up	% aligned
SRR006150			
BWA default	11 644.02		64.60
BWA @ SRmapper def	14 794.12		73.34
SRmapper @ BWA def	2706	4.30	59.44
SRmapper default	3552	4.17	67.25
SRR020477			
BWA default	1547		55.76
BWA @ SRmapper def	8192		68.43
SRmapper @ BWA def	676	2.29	54.90
SRmapper default	973	8.42	71.53
SRR539393			
BWA default	2795		84.18
BWA @ SRmapper def	11 248		88.56
SRmapper @ BWA def	1317	2.12	82.72
SRmapper default	1542	7.29	89.29

Alignment time for 13.14 million pairs of 51-bp reads (SRR006150), 2.04 million pairs of 76-bp reads (SRR020477) and 2.25 million pairs of 100-bp reads (SRR539393) downloaded from the sequence read archive to the human genome was determined. BWA aln was run as described in Table 1, and BWA sampe was run with the –A option to disable Smith–Waterman alignment. Default mismatches allowed for SRR006150 are 3 and 12 for BWA and SRmapper, respectively, for SRR020477 are 4 and 28 for BWA and SRmapper, respectively, and for SRR539393 are 5 and 40 for BWA and SRmapper, respectively.

3.3 Results on simulated data

To further compare the accuracy of SRmapper with BWA, simulated single-end reads of length 50 and 100 bp were generated from the human genome. To model reads from a sequencer, the simulated reads were allowed to have sequencer errors as well as single-nucleotide polymorphisms (SNPs) and indels. All indels were of length 1–5. Sequencer error rate was set at 1.5%, and SNP and indel rates were set at 0.09% and 0.01%, respectively, as in the SNP and indel rates in the original testing of BWA (Li and Durbin, 2009). Tests were performed using both BWA's default mismatch parameter and SRmapper's default mismatch parameter. As both tools randomly choose an alignment for reads with multiple equally best hits and report zero confidence, only reads that had a non-zero quality score were considered for determining accuracy. Accuracy here refers to simulated reads that were aligned back to the locations of the genome from which they were derived.

For all of the test runs, both alignment tools had high accuracies for reads confidently aligned (Table 3). BWA did show a higher accuracy across all tests, but as read length increased, incorrectly placing confident reads became less of an issue with SRmapper, having an error rate of ~1:250 confident reads being incorrect. It is noteworthy that in reviewing reads that were incorrectly placed, the main cause of incorrect placement of reads was because of SNPs or sequencer errors, which resulted in the read aligning with a higher quality to an alternative location in the genome. However, this issue is better considered to be a weakness of the sequencers and reference assembly than an error of the alignment program, as there is not always sufficient information in short reads to properly determine where a read should be placed when there are differences between the reference and the genome to be assembled. Likewise, 'incorrect' placement of reads because of sequencing errors should be considered a weakness of sequencers and not the fault of the alignment programs. To further evaluate the accuracy of SRmapper in relation to BWA, simulated pair-end reads were created using wgsim (https://github.com/lh3/wgsim). The receiving operating characteristics (ROC) curves show that for this dataset, BWA has a lower error rate and a somewhat higher sensitivity for the data set shown (Supplementary Fig. S4). The error rate of SRmapper for pair-end reads would be expected to be lower than for single-end reads, as for a pair-end read to be misaligned, both mates in the pair would have to be incorrectly aligned to the same area. Without Smith–Waterman alignment, SRmapper would be expected to be somewhat less sensitive in pair-end alignment than single-end alignment, as both ends of a pair must be aligned to generate a pair-end alignment.

Table 3. Evaluation of the accuracy of SRmapper on simulated reads

Simulated reads	50 bp % aligned	50 bp % correct	100 bp % aligned	100 bp % correct
BWA @ def	83.55	99.62	87.30	99.90
BWA @ SRmapper	83.96	99.58	87.88	99.82
SRmapper @ BWA	81.83	98.79	88.07	99.58
SRmapper @ def	82.30	98.32	89.06	99.19

A total of 100 000 simulated reads were generated from the human genome with a 1.5% sequencer error rate, a 0.09 SNP rate and a 0.01 insertion–deletion rate. Insertions and deletions were of random size between 1 and 5 bp. SRmapper and BWA were run as described in the legend of Table 2. The %aligned is the per cent of reads confidently aligned, and %correct is calculated as (confident reads correctly aligned)/(reads confidently aligned) × 100.

4 CONCLUSIONS

Here, we have implemented a reference genome-hashing alignment tool that shows a significant speed increase over BWA while maintaining similar alignment rates and a comparable memory footprint. Additionally, SRmapper is less sensitive to increasing the number of mismatches allowed in short reads than BWA is, as the speed advantage SRmapper possesses over BWA increases as the number of mismatches allowed increases. Also, pair-end alignment against reads for the human genome can be performed on a PC with as little as 4 GB of memory, whereas pair-end alignment with BWA requires a computer with a larger amount of memory. Although SRmapper does not possess all the features of BWA with no gapped alignment or Smith–Waterman alignment for pair-end reads being the two most notable, its capabilities are sufficient to demonstrate that hash table-based alignment tools can be created with similar sensitivities and memory requirements and a higher speed than the current BWT alignment tools. The ROC curves show that a large portion of the false alignments occurs with those having few or no mismatches. The only way this can occur is in repetitive regions of the genome where all possible alignments may not be found. Currently, SRmapper does not retain suboptimal alignments or use them for determination of

quality in alignments that explains why SRmapper has a higher rate of false-positive results than BWA. This is a significant advantage BWA currently has over SRmapper and many other popular alignment tools (http://lh3lh3.users.sourceforge.net/alnROC.shtml).

An additional advantage SRmapper possesses over the BWT tools is that its alignment need not use the whole read simultaneously. In theory, this property would allow SRmapper to hunt down chromosomal translocations and deletions longer than those currently detectable by alignment tools. In detection of a read containing a long deletion or translocation, alignment would begin from one end of the read proceeding until a stretch of bases with poor correspondence to the reference was found. Alignment would then commence from the other end of the read and proceed to where the first alignment began showing poor correspondence with the reference.

As different alignment tools have different pros and cons, using several alignment tools in sequence could provide a better balance among speed, sensitivity and accuracy. For example, as SRmapper is fast, one can use it first to align reads. Unaligned reads and reads that are more likely to be misaligned can then be evaluated by other aligners, such as BWA.

ACKNOWLEDGEMENT

The authors thank the University of Missouri Bioinformatics Consortium for providing computational resources.

Conflict of Interest: none declared.

REFERENCES

David,M. *et al.* (2011) SHRiMP2: sensitive yet practical short read mapping. *Bioinformatics*, **27**, 1011–1012.

Ewing,B. and Green,P. (1998) Base-calling of automated sequencer traces using phred. II. *Error probabilities*. *Genome Res.*, **8**, 186–194.

Ewing,B. *et al.* (1998) Base-calling of automated sequencer traces using phred. I. *Accuracy assessment*. *Genome Res.*, **8**, 175–185.

Homer,N. *et al.* (2009) BFAST: an alignment tool for large scale genome resequencing. *PLoS ONE*, **4**, e7767.

Langmead,B. *et al.* (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li,H. and Durbin,R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, **26**, 589–595.

Li,H. *et al.* (2009a) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Li,R. *et al.* (2009b) SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, **25**, 1966–1967.

Li,Y. *et al.* (2011) WHAM: A high-throughput sequence alignment method. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011*. Athens, Greece, pp. 445–456.

Lunter,G. and Goodson,M. (2011) Stampy: a statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Res.*, **21**, 936–939.

Ng,P.C. and Kirkness,E.F. (2010) Whole genome sequencing. *Methods Mol. Biol.*, **628**, 215–226.

Ning,Z. *et al.* (2001) SSAHA: a fast search method for large DNA databases. *Genome Res.*, **11**, 1725–1729.

Pareek,C.S. *et al.* (2011) Sequencing technologies and genome sequencing. *J. Appl. Genet.*, **52**, 413–435.

Rafnar,T. *et al.* (2011) Mutations in BRIP1 confer high risk of ovarian cancer. *Nat. Genet.*, **43**, 1104–1107.

Rothberg,J.M. *et al.* (2011) An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, **475**, 348–352.