Alignment of Next-Generation Sequencing Reads

Knut Reinert,[1] Ben Langmead,[2] David Weese,[1] and Dirk J. Evers[3]

[1]Department of Mathematics and Computer Science, Freie Universität Berlin, 14195 Berlin, Germany; email: knut.reinert@fu-berlin.de, david.weese@fu-berlin.de

[2]Department of Computer Science and Center for Computational Biology, Johns Hopkins University, Baltimore, Maryland 21218; email: langmea@cs.jhu.edu

[3]Molecular Health GmbH, 69115 Heidelberg, Germany; email: dirk.evers@molecularhealth.com

Annu. Rev. Genomics Hum. Genet. 2015. 16:133–51

First published online as a Review in Advance on May 4, 2015

The *Annual Review of Genomics and Human Genetics* is online at genom.annualreviews.org

This article's doi: 10.1146/annurev-genom-090413-025358

Keywords

high-throughput sequencing, read mapping, string indices

Abstract

High-throughput DNA sequencing has considerably changed the possibilities for conducting biomedical research by measuring billions of short DNA or RNA fragments. A central computational problem, and for many applications a first step, consists of determining where the fragments came from in the original genome. In this article, we review the main techniques for generating the fragments, the main applications, and the main algorithmic ideas for computing a solution to the read alignment problem. In addition, we describe pitfalls and difficulties connected to determining the correct positions of reads.

INTRODUCTION

The advent of next-generation sequencing (NGS, also known as high-throughput sequencing or deep sequencing) in 2005 led to an explosion in the amount of sequencing data. A typical sequencing experiment now yields billions of snippets of DNA characters (reads) that are sampled from the DNA molecules in a biological specimen. Modern microarrays, by contrast, produce only on the order of millions of decimal intensity values per experiment. Because of this explosion in the number of sequences, a large fraction of the computational effort spent analyzing biological data is now dedicated to determining where these reads came from in the sample and how they fit together. As more reads are traced back to their original locations, a clearer picture emerges of which DNA sequences are present in the biological specimen and in what abundance. This knowledge then enables investigators to address relevant scientific questions, such as how a particular genome differs from a reference genome, or which genes or isoforms are differentially expressed between two conditions.

In the early days of NGS and sequencing by synthesis, researchers realized that the sequence alignment software tools popular at the time [e.g., Basic Local Alignment Search Tool (BLAST) (2)] were simply not efficient enough to analyze NGS-scale data sets, and were also not engineered for the problem. Since then, a large number of publications have described new algorithms and methods to solve this problem. These efforts originated from various fields of study—often computer science, but sometimes statistics and mathematics. Some methods are geared toward certain sequencing technologies, whereas others are more general. Some primarily address efficiency, whereas others address scalability, accuracy, or interpretability. In this review, we sift through and summarize this large body of work and distill essential points relevant to a wide audience of both method developers and practitioners. We focus on read alignment, which is often the most demanding computational problem tackled in sequencing studies.

Solving the read alignment problem opens up a multitude of possible applications. Experiments requiring read alignment solutions are manifold. The classic application is to align sequencing reads to a high-quality reference genome, such as the human genome reference or a bacterial genome reference. In many ways, NGS short-read technology became viable only after the Human Genome Project and the completion of model organism references over the last decade. The main goal in mapping genomic DNA back to the organism's genome reference sequence is to determine sequence variation. Variations range from single point mutations to short insertions or deletions (indels) to even larger-scale complex variations spanning thousands of nucleotides or more. Applications in this space include variant detection in the germlines of individuals, families, or populations to ascertain frequent and infrequent variation—such as disease-causing mutations—within the gene pool.

DNA sequencing of tumor material has become feasible through the massive drop in sequencing cost. It is now possible to produce data from a few nanograms of pure DNA. Standard solid-tumor histopathology practice typically results in DNA being extracted from formalin-fixed, paraffin-embedded (FFPE) tissue. DNA fragments derived from this archival format are limited in length and quality, making short to medium-length sequencing methods well suited for this purpose. The surrounding normal tissue or a separate healthy specimen from the same individual is frequently used to differentiate somatic from germline variation. Analysis can also be performed on tumor data only. However, differentiating germline from tumor variants can become an issue in this case. The variation frequency in a population of tumor cells down to a detection limit of a few percent can be measured if the depth of coverage (i.e., oversampling) is sufficient. Copy-number variation may be detected at very high resolution across both whole human genomes and whole-exome samples. To enable this type of application, the observed oversampling rate of a given genomic region is compared with the expected rate to infer copy-number changes. Various biases introduced through the sample preparation process, as well as systematic effects such as Poisson sampling, must be taken into account when counting the mapped fragments.

RNA reverse transcription and cDNA sequencing with alignment of the resulting reads are performed to measure expression levels of transcripts in a specific tissue. Here, alignments must be split along the exon-intron boundaries; very long introns and the small sizes of some exons make this a challenging problem to solve. In cancer, sequencing of transcripts can also identify and verify expressed somatic variations, such as fusion genes or single point mutations in tissue-specific isoforms. Expression levels can also be derived from this form of analysis.

Some applications can be performed significantly faster if run via specialized alignment programs. For example, DNA fragments from ChIP-seq (chromatin immunoprecipitation with DNA sequencing) experiments need to be aligned to a reference genome. The use of a specialized alignment program can accelerate this process, for instance, by aligning only against regions of interest.

Epigenomic analysis is another application that requires specialized alignment algorithms because the alphabet of nucleotides must be extended to take into account the methylation modifications of nucleotides. In stark contrast, the standard sample preparation technique of bisulfite treatment of DNA produces a reduced-complexity alphabet. This converts unmethylated cytosines to uracil, which in turn is polymerase chain reaction (PCR)–amplified as thymine, resulting in an increased number of mismatches that need to be taken into account during alignment to give adequate sensitivity.

Another area that requires specialized alignment programs is metagenomics. Here, the problem is to identify the organism or phylogenetic group that gives rise to a DNA fragment taken from a sample containing multiple organisms. Additionally, the position of the read within the genome and the putative function of the genomic region at that position may be ascertained. Example experiments involve samples taken from human skin, body cavities, or digestive and respiratory tracts.

The problem of read alignment is also closely connected to de novo assembly. For example, say an investigator has a large collection of reads and would like to piece them back together, thereby re-creating the sequences of the molecules from which they originated. Without any further information, the only possibility is to assemble the reads de novo. The first step might be to identify similarities between reads; for example, if the last several nucleotides of one read are similar to the first several nucleotides of another, then the reads may have originated from overlapping stretches of the same molecule. An algorithm can be used to find the assembled sequence most consistent with all of the observed overlaps. Many methods for de novo assembly have been proposed, and it is an active area of study today (5, 7, 48, 56).

That said, there are many species for which a high-quality assembled genome sequence already exists, including humans and most model species. Because two individuals of the same species have extremely similar genome sequences (e.g., two unrelated humans have genomes that are approximately 99.8% similar by sequence), one can use an assembled genome of the appropriate species as a template or guide to assist in piecing reads together. An assembled genome sequence used in this way is called a reference genome, or simply a reference. When a reference genome is available, one can proceed by taking each read and determining where the read sequence most closely matches the reference genome sequence, which then represents the best guess as to where the read originated with respect to the reference. (We elaborate below on what we mean by "most closely.")

Sequencing Technologies

In this section, we explore in greater depth how reads are obtained. Several technologies are on the market, and more are on the horizon. The introduction of NGS has led to a dramatic increase in

Table 1 Approximate run times, yields, read lengths, and sequencing error rates of different high-throughput sequencing technologies as of mid-2014 (22)

Technology	Instrument	Run time	Yield (Mb/run)	Read length (bp)	Error rate (%)
Sanger	3730xl (capillary)	2 h	0.08	∼1,000	0.1–1
Illumina	HiSeq 2500	6 days	1,000,000	2 × 125	≥0.1
SOLiD	SOLiD 4	12 days	50,000	35–50	>0.06
454	FLX Titanium	10 h	500	400	1
SMRT	PacBio RS	0.5–2 h	500	∼10,000	16
Ion Torrent	Ion Proton 318	7 h	2,000	400	1

Abbreviations: bp, base pairs; SMRT, single-molecule real-time.

sequencing throughput over the last few years. This technology allows the production of billions of base pairs (bp) per day in the form of reads of length 100 bp or more. Since 2005, when 454 Life Sciences released the first commercially available high-throughput sequencing machine, throughput has continued to increase, and new technologies will provide longer reads than are currently available. Moreover, sequencing costs are decreasing more rapidly than the costs for hard disk storage or Moore's law for computing costs (62).

Currently available high-throughput sequencing platforms include SOLiD (Life Technologies Corporation), Illumina (Illumina Inc.), Ion Torrent (Life Technologies Corporation), single-molecule real-time (SMRT) (Pacific Biosciences Inc.), and 454 (Roche Diagnostics Corporation). **Table 1** compares the run times, yields, read lengths, and sequencing error rates of these platforms. Compared with older sequencing technologies by gel electrophoresis, a key improvement is to cycle and image the incorporation of nucleotides or to detect the incorporation in real time. Removing the need for a gel tremendously reduced sequencing costs and has made it possible to miniaturize and parallelize sequencing. Common to all technologies is that the DNA is first fractionated into smaller double-stranded fragments, which are optionally amplified and then sequenced in parallel from one or both ends. The sequencing process is either (*a*) cycled, where a single terminated base (Illumina), a single primer (SOLiD), or a run of a single nucleotide (454) is incorporated, or (*b*) carried out in real time (SMRT, Ion Torrent). In the case of the technologies that incorporate one nucleotide or primer at a time (Illumina, SOLiD), the color of the signal determines the nucleotide or primer integrated in a given cycle. For technologies that incorporate an entire homopolymer run at a time (454, Ion Torrent, SMRT) the intensity or length (SMRT) of the signal determines the length of the run.

Cycled technologies amplify the DNA fragments within small clusters (Illumina, SOLiD, 454) and require that each cluster contain thousands of identical molecules that are sequenced synchronously and whose combined signal is measured. Synchronization is realized by using reversible terminator chemistry, which ensures that only one nucleotide is incorporated at a time (Illumina); by performing ligation of only one primer at a time (SOLiD); or by washing only one nucleotide over all fragments, which results in the incorporation of only one homopolymer run at a time (454).

One type of error in technologies with a prior amplification step is the so-called dephasing, where because of a synchronization loss in a cluster (e.g., caused by missed incorporations) the signals of the current base and its neighbors in the DNA template interfere and lead to an increase of miscalls toward the ends of the reads (see **Figure 1*a***). Another type of error is related to a limitation in signal resolution: Technologies that use signal intensity (454) or length (SMRT, Ion Torrent) to determine the length of homopolymer runs cannot reliably detect the length of large runs owing to a limited sensor resolution and an increase of noise (e.g., caused by variations of the

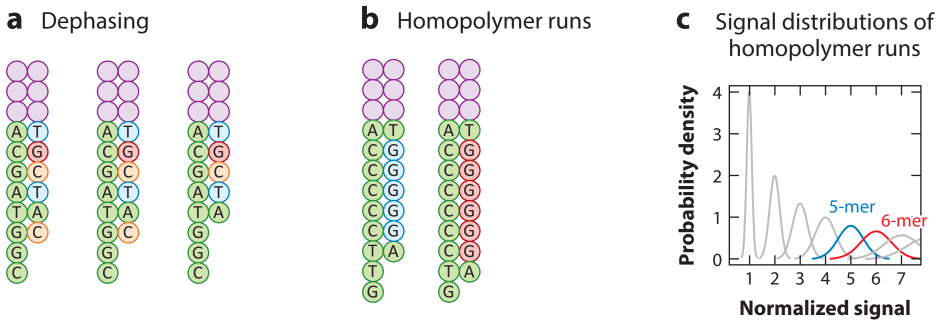


Figure 1

(*a*) Dephasing, in which missed nucleotide incorporations or failed terminator cleavage contributes to a loss of synchronization. (*b*) 454 sequencing, in which a whole homopolymer run is incorporated at a time. (*c*) Signal distributions in homopolymer runs. The intensity of the emitted signal follows a normal distribution, with a mean and deviation proportional to the homopolymer length (44).

DNA polymerase speed). Such technologies typically produce reads with indels in homopolymer runs. A third type of error causes deletions of bases in technologies that omit a prior template amplification (SMRT, Ion Torrent) owing to weak and undetected signals. However, errors in the sequencing process can be discerned by using base-call quality values and redundancy, such as a high base coverage (e.g., 20 and higher) (14).

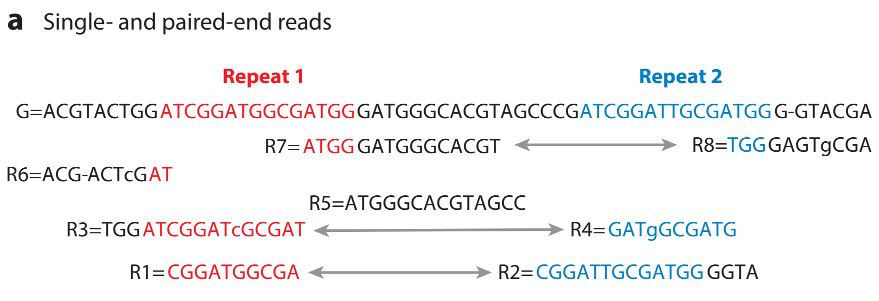
As discussed in more detail below, it is advantageous to generate reads that are as long as possible in order to resolve ambiguity during the read alignment. However, all of the above-mentioned technologies have limitations in producing very long single-end reads. To address this problem, many technologies have protocols to produce pairs of reads. To generate paired-end reads, the sequencer reads a string of nucleotides from both ends of a longer fragment. In a typical experiment using an Illumina HiSeq instrument, fragments are 250–500 bp long, and the strings of nucleotides read from either end are approximately 100–150 bp long. The major sequencing instruments from Illumina, Life Technologies, and Roche, for example, have protocols for yielding paired-end reads.

Approximate String Matching Problem

As described above, the sequencing technology determines some parameters that have a bearing on the read alignment problem, such as read length and error rates. Nevertheless, for each read (single ended or paired), the goal is to find its true location with respect to the reference. The true location is not known beforehand and is found by solving an approximate matching problem—that is, searching for occurrences of the read sequence within the reference sequence but allowing for some mismatches and gaps between the two.

If a genome had no repeats and a sequencing experiment introduced no errors, then the need for approximate matching would disappear. That is, assuming a sufficient read length relative to the genome size, one could find the true locations using exact matching. Unfortunately, neither assumption holds: Eukaryotic genomes are rich in repeats, and each sequencing technology introduces errors, as outlined above. If a repeat sequence is perfectly identical, then it is not possible to determine a read's true location if it lies completely within the repeat.

This is not always the case (as depicted in **Figure 2**, where two repeats differ in one base). Because the reads can also contain errors, one must resort to approximate string matching. The



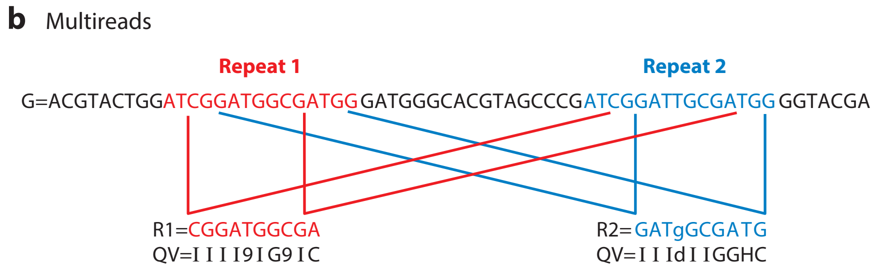


Figure 2

(*a*) A small example of a genome containing two repeat instances that differ in one base, along with sampled single-end reads (R5 and R6) and paired-end reads (R1-R2, R3-R4, and R7-R8). Sequencing errors are shown as lowercase letters; mate pairs are connected by arrows. Note that insertions (see read R8) and deletions (see read R6) are possible. (*b*) The same example genome with multireads (R1 and R2). Both reads map to two different genomic locations with up to one error, and therefore both matches are valid. The colors of the reads indicate their true locations. Read R1 maps with no errors to its true location and one error to the second valid location, whereas read R2 maps with one error to its true location and no error to the other valid location. A quality value (QV) string is included below each read, where each character of the string encodes the probability that the acquired base is an error.

rationale would be to choose an error model such that the location where the read aligns with the minimal number of errors is likely to be the true location. Although this can be wrong (see **Figure 2**), it is a reasonable approximation.

The error model must be chosen according to the sequencing technology (e.g., Illumina reads have more mismatches than indels at a relatively low error rate, whereas SMRT reads have a large absolute number of indels and a high error rate) and according to the characteristics of the data at hand. For example, mapping to a reference genome from a non-closely-related strain requires accommodating for more differences between read and reference.

In general, the most widely used error models are the Hamming distance, which accommodates only for mismatches between the read and a chosen genomic location, and the edit distance, which accounts for mismatches and indels. An alternative to using the edit distance, which treats all insertions, deletions, and mismatches equally, is to use the weighted edit distance, which assigns weights to errors to distinguish between mismatches and indels, and can even weight position-specific errors differently by taking into account the quality values that represent the probability of the base call being an error (**Figure 2*b*** includes two such quality strings). Other error models, such as affine gap costs for longer indels, are also possible at the expense of a higher compute cost. In the following section, we review the main algorithmic ideas on a high level, concentrating on the model described above for aligning the sequencing reads completely to the reference. There are variations of the problem (e.g., allowing the ends of reads to be unaligned, or, for RNA-seq reads, allowing parts of a read to match nonconsecutively), but overall they use similar approaches.

MAIN ALGORITHMIC IDEAS

A brief review of the literature will quickly reveal a standard solution for approximate string matching with indels. The problem can be solved for one read of length m and a genome of length n by using a dynamic programming algorithm. This kind of algorithm uses an amount of time and memory that grows proportionally to $m \times n$. Some algorithms can reduce the memory requirement to grow proportionally to m, but the time required is still proportional to $m \times n$, which is much too slow to align billions of reads to genomes of size 10^9 bp or more.

There are two main algorithmic ideas to address the problem of large input sizes (both in number of reads and size of the reference) for approximate string matching: filtering and indexing. Filtering approaches quickly exclude large regions of the reference where no approximate match can be found. This can, for example, be done by identifying short regions in the reference (also known as k-mers) that share a short piece of the read without errors, often called a seed. Regions that do not share such a short region are filtered out. In addition to seeding filters, there are filters based on shared q-gram counts (8) or based on the pigeonhole lemma.

Indexing strategies involve preprocessing the reference sequence, the set of reads, or both in a more intricate way. A benefit of such preprocessing into string indices is that it typically does not require scanning the whole reference, and it can therefore conduct queries much faster at the expense of larger memory consumption. The string indices that are currently used are the suffix array (42); the enhanced suffix array (1); and the FM-index (17), a data structure based on the Burrows-Wheeler transform (9) and some auxiliary tables. The enhanced suffix array and the FM-index are capable of determining whether a query sequence occurs in the reference in linear time with respect to the length of the query. The FM-index makes very economical use of memory, but it is still quite fast to query in practice. The suffix array is larger but can also be somewhat faster to query.

Finally, most algorithms—whether they incorporate filtering, indices, or both—require a final verification to ensure that the read has an approximate occurrence in the used error model (e.g., has fewer than three indels). This is usually done by fast versions of dynamic programming–based algorithms that can be sped up either by computing only a certain region of the dynamic programming matrix [banded alignment (e.g., 46)] or by using bit-vector operations to achieve some parallelism [e.g., the Myers bit-vector algorithm (47)].

Filters

To reliably determine which region of the reference can be discarded because it cannot contain an approximate match, filters typically use one of two lemmata: the pigeonhole lemma or the q-gram lemma. Both determine a minimal length or number of exact q-grams that a read with a certain number of errors shares with the reference sequence.

Assume that one wants to find all approximate matches of a read with at most k errors. In its simplest form, the pigeonhole lemma states that if the read is cut into $k + 1$ pieces (also called seeds), then at least one piece occurs without error in an approximate match (4). This lemma is simply a result of the observation that k errors cannot modify more than k of the pieces. A pigeonhole filter therefore divides each read into $k + 1$ nonoverlapping seeds of (almost) equal size and, in parallel, searches them in a scan over the reference. **Figure 3*a*** shows an example of a pigeonhole filter approach. If all seeds have the same size q, a q-gram index allows the efficient

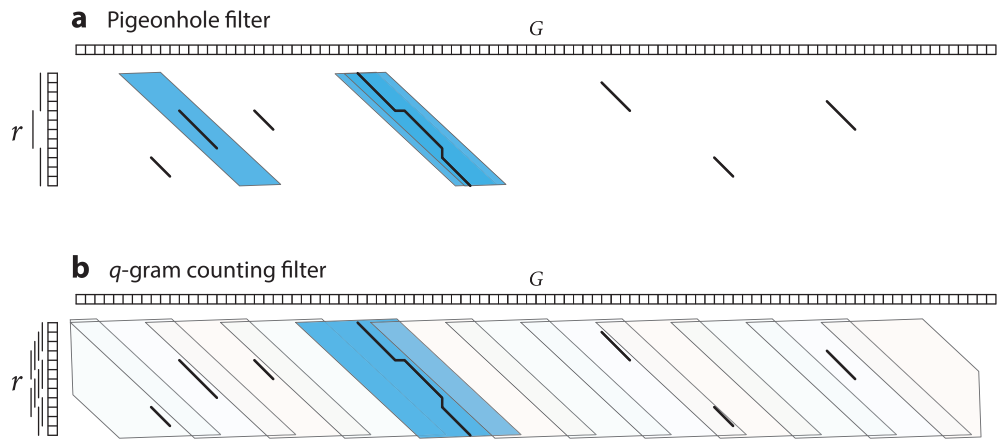


Figure 3

The two types of *q*-gram filters. (*a*) The pigeonhole filter divides the read into nonoverlapping *q*-grams (seeds) and reports for each occurrence in the reference *G* a surrounding parallelogram as a candidate region. (*b*) The *q*-gram counting filter counts common *q*-grams in overlapping parallelograms. Parallelograms with a sufficiently high number of common *q*-grams are reported as candidate regions.

storage and retrieval of all seeds matching the *q*-gram that slides over the reference sequence. Some read mappers use variants of the pigeonhole lemma—for example, dividing the reads into more seeds (e.g., $k + 2$) and searching for at least two matching seeds (e.g., Illumina's original ELAND algorithm and References 28, 36, and 37), or dividing the reads into fewer seeds while tolerating one or two errors in them (32, 60). Others allow gapped seeds with an adaptive length (30). Seeds with errors are typically searched using a backtracking approach in a string tree (e.g., the FM-index described in the next section).

The *q*-gram lemma considers all overlapping *q*-grams of a read and gives a lower bound for the number of *q*-grams that a *k*-error match in the reference shares with the read. This number is determined by the worst case, where *k* errors are equidistantly distributed along a read of length *n* and modify at most *kq* of the $n - q + 1$ overlapping *q*-grams (**Figure 4**). Hence, the lower bound is $t(n, k, q) := n - (k + 1)q + 1$, the number of intact *q*-grams remaining. A *q*-gram counting filter searches for genomic regions where at least $t(n, k, q)$ *q*-grams of a read can be found and reports them as a potential match. All other regions can be safely discarded. The reference is divided into overlapping regions, with a counter for each read and region. The counters are updated while sliding a *q*-gram over either the reads or the reference that is simultaneously searched in the other using a *q*-gram index. The existing counting filters differ only in the shape and size of the region and details in counting (8, 11, 50, 53, 66, 67). **Figure 3*b*** shows the counting filter used by RazerS 3 (67), where regions are parallelograms and the sliding reference *q*-gram is searched in the reads.

Some recent read mappers use alternative pigeonhole filters (23, 24, 60). Instead of searching common seeds by sliding a *q*-gram over either the reads or the reference and searching it in the other using a *q*-gram index, two indices are built (one over the reads and one over the references) and searched in parallel for occurrences of the same *q*-gram.

Indices

The simplest but most commonly used index for read mapping is the *q*-gram index (also known as the *k*-mer index). A *q*-gram is simply a sequence of *q* characters over a given alphabet Σ, e.g.,

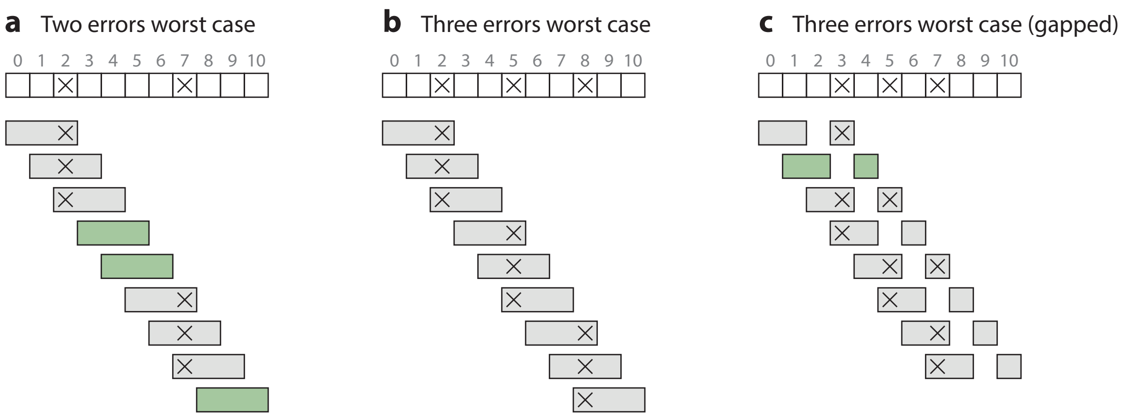


Figure 4

Mismatch patterns of length 11. (*a*,*b*) These panels show worst cases of the *q*-gram lemma where a maximal number of ungapped 3-grams are modified. (*c*) For three mismatches, a gapped shape should be preferred because it yields a higher threshold.

$\Sigma = \{\mathtt{A,C,G,T}\}$ in the case of DNA. Given a text consisting of a single sequence or a set of sequences, for each possible *q*-gram a *q*-gram index allows the quick retrieval of all its occurrences in the text; it is therefore perfectly suited for the two types of *q*-gram filters described above. An optimal *q*-gram index determines whether a *q*-gram occurs in the text in constant time and retrieves its occurrences in time proportional to the number of occurrences. It can be implemented using two tables: an occurrence table to store the occurrences grouped by *q*-gram, and a lookup table to map each *q*-gram to its first entry in the occurrence table. For example, in a single sequence of length n, the occurrence table has $n - q + 1$ different entries between 0 and $n - q$, one for each position a *q*-gram can start at. The lookup table typically occupies an entry for each of the $|\Sigma|^q$ possible *q*-grams. To look up the occurrences of a *q*-gram, it is first converted into a number c between 0 and $|\Sigma|^q - 1$ (e.g., in the case of DNA, by interpreting its characters as digits of a base-4 numeral system). The lookup table entry at position c is then used to determine the first entry of the group of occurrences in the occurrence table. Assuming that occurrences are grouped in the same order as the *q*-grams are mapped to numbers, the end of the group is stored at entry $c + 1$ in the lookup table. **Figure 5** gives a detailed example of such a *q*-gram index. If the parameter q becomes too large (e.g., >16 for DNA), then the space consumption of the lookup table becomes prohibitive, and one must resort to hashing techniques (e.g., 49).

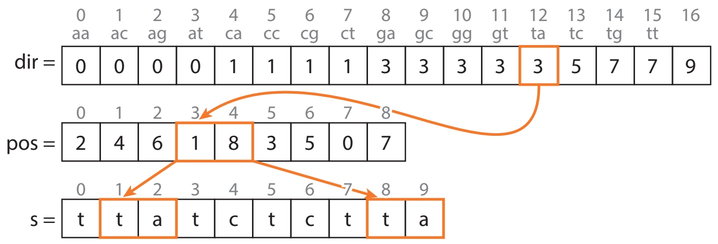


Figure 5

A 2-gram index of `ttatctctta`. To look up all occurrences of the 2-gram `ta`, one first determines its numerical value code, `(ta)` = 12. At positions 12 and 13, the lookup table "dir" stores the boundaries 3 and 5 (excluding) of the group of occurrences in the occurrence table "pos." Therefore, positions 1 and 8 are the only occurrences of `ta` in the text.

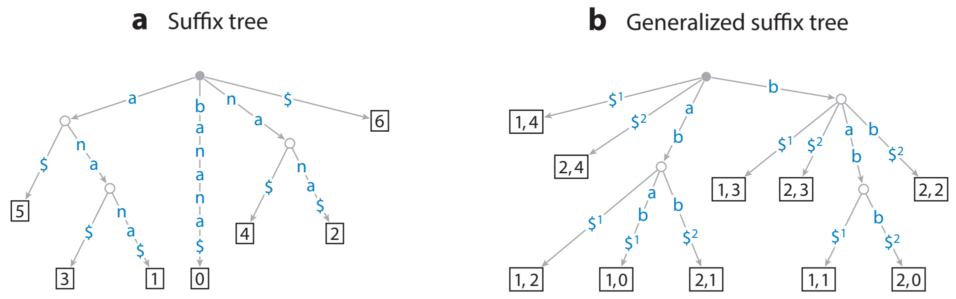


Figure 6

(*a*) The suffix tree of the single sequence `banana$`. (*b*) The generalized suffix tree of the two sequences `abab$`[1] and `babb$`[2]. Leaves are labeled with string positions i or (j, i), representing suffixes of the jth sequence starting at position i.

Whereas the q-gram index is typically used in filters to look up exact subsequences of a fixed length q, another type of index is used in more recent read mappers to directly search reads with errors: the so-called suffix tree, a powerful data structure (68). The suffix tree of a given text represents all suffixes of the text, allows one to search for all occurrences of a string (i.e., a read) in optimal time, and can itself be built in linear time.

In its original definition, the suffix tree is a rooted tree whose edges are labeled with strings over an alphabet Σ. Each internal node is branching—that is, has at least two children—and outgoing edges begin with distinct characters. It is called a suffix tree because each path from the root node to a leaf spells out a suffix of the text and vice versa. **Figure 6** shows an example of a suffix tree for a single sequence (`banana$`) and the generalization to multiple sequences. (The artificial \$ character must be appended to each sequence to let all suffixes end in leaves.) In contrast to the q-gram index, a suffix tree can be used to search for sequences of arbitrary length in a given text. To search for a sequence, the suffix tree is traversed from the root along the path (if one exists) that spells out the sequence sought. If the sequence occurs in the text, then that path ends either in a node or on an edge. In the latter case, the descent proceeds to the next lower node. The occurrences in text are then represented by the leaves below that node. If, for example, `an` is sought using the suffix tree of `banana$` in **Figure 6*a***, then the edges `a` and `na` are descended. The path would end in the middle of the `na` edge and is thus extended to the node at its end. The leaves below that node are 3 and 1, the start positions of occurrences of `an` in `banana`.

Since the suffix tree was introduced (68), many different implementations have been published because the original suffix tree needs a lot of space. The suffix tree also suffers from poor locality of reference; that is, it cannot make effective use of the fast cache memories available on typical processors. For these reasons, it has often been replaced by the suffix array (42), which originally traded lower memory requirements against a suboptimal search time. The most recent improvements could overcome those limitations. The enhanced suffix array (1) and the FM-index (17) are both used for read mapping (25, 32–34, 38, 60). Although the enhanced suffix array is the fastest implementation for many suffix tree applications, the FM-index has a much smaller memory footprint while still being very fast and is therefore the current method of choice.

As described above, a suffix index (such as a suffix tree) organizes all of the suffixes of a text in a concise way. This text could be, for example, the entire GRCh37 build of the human reference genome. Equipped with the index, one can conduct a search for approximate matches of a sequence read in a reference genome by considering strings that are "near" (i.e., similar to) the read string. (For instance, the misspelled word "rectongle" will not appear in an English dictionary, but the

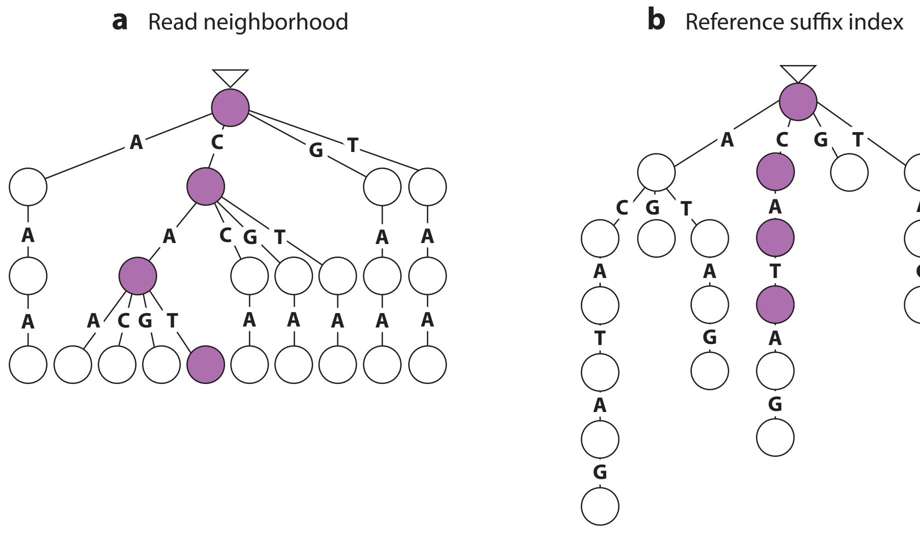


Figure 7

(*a*) A neighborhood tree representing the strings that are near the read string, and (*b*) a tree representing suffixes of the reference genome. Finding approximate matches of the read in the genome involves cotraversing these two trees and looking for paths such as the one highlighted in purple. This path corresponds to a one-mismatch alignment of the read `CAA` to the substring `CAT` of the reference genome.

nearby word "rectangle" will.) Some popular read alignment tools employ a method related to this idea, using the index to rapidly identify occurrences of strings that are near the query (33, 34, 38). More formally, this search can be cast as a cotraversal of two tree data structures; one tree corresponds to the suffix index of the reference genome, and the other corresponds to the neighborhood of strings that are near the read string. **Figure 7** depicts this in more detail.

Verification

In the second step, a verification algorithm examines the area around each candidate to determine whether a full high-scoring alignment exists in that vicinity. In many aligners, this verification step is the bottleneck, so efficient methods for verification are desirable.

In the simplest case, when no gaps are considered, candidates can be verified by comparing the read against the reference while counting the number of mismatches. For verification with gaps, Myers (47) proposed an algorithm that exploits bit parallelism to efficiently compute the edit distance. Both the original algorithm and an even more efficient banded variant have been used (60, 66, 67). For more flexible error models, the local alignment algorithm developed by Smith & Waterman (61) can be used, e.g., to incorporate base-call qualities at mismatching bases, as described at the beginning of this article. Recent mappers (11, 32, 53) use single-instruction multiple-data (SIMD) vectorized variants of that algorithm (e.g., as proposed in 15).

Aligning Read Pairs

Given a read that originates from a repetitive portion of the genome, a read aligner often cannot say for certain which copy of the repeat the read came from. This leads to low mapping quality (discussed below) and potential mistakes in downstream analyses. To combat this ambiguity, one

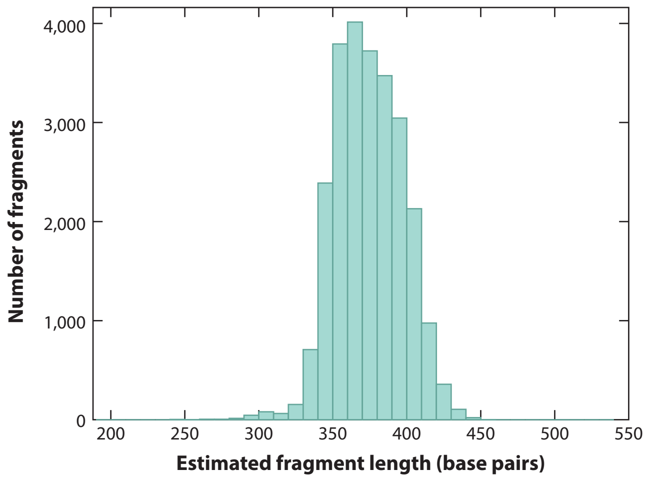


Figure 8

A fragment length distribution as estimated from a collection of paired-end Illumina reads derived from human DNA. The data set is accession ERR037900 in the Sequence Read Archive.

might simply make the reads longer. Longer reads are more likely to span nonrepetitive sequence, providing a unique anchor with which to resolve the read's true origin. But generating longer reads is often not possible given the constraints of sequencing technology. Most high-throughput instruments currently available from Illumina, for example, generate reads that are up to 150 bp long.

Instead of making the reads longer, it is helpful to generate reads in pairs, called paired-end reads (see above). When aligning a paired-end read, the aligner can use the fact that the paired sequences were derived from either end of a fragment of (approximately) known length. The length is approximate because the laboratory step that isolates fragments of the desired length is not perfect; rather, it yields a distribution of fragment lengths, which can be characterized by a histogram or by a parametric distribution such as a Gaussian function. **Figure 8** shows a typical distribution of fragment lengths.

Paired-end reads can be more effective than unpaired reads at resolving repetitive portions of the genome. This is because the true location for an end that originated from inside a repeat can be resolved as long as the other end originated from a nonrepetitive portion of the genome and can be assigned to its true location. **Figure 9** illustrates a few scenarios where paired-end reads can and cannot help to resolve the true point of origin for reads originating within or near repeats.

When an aligner aligns a paired-end read, it prefers to align the two ends in a way that is compatible with both having originated from either end of a fragment. For instance, the fragment length during the sample preparation should be approximately 300 bp. The first end might align equally well to hundreds of locations across the reference. But say that the second end aligns uniquely to a single location that is within 300 bp of one of the alignments for the first end. Because both ends originated from the same fragment, one can deduce with high confidence that the correct alignment for the first end is the one that lies within 300 bp of the unique alignment for the second end.

WHAT CAN GO WRONG?

In the following sections, we outline situations where read aligners can make mistakes or otherwise create challenges for interpretation.

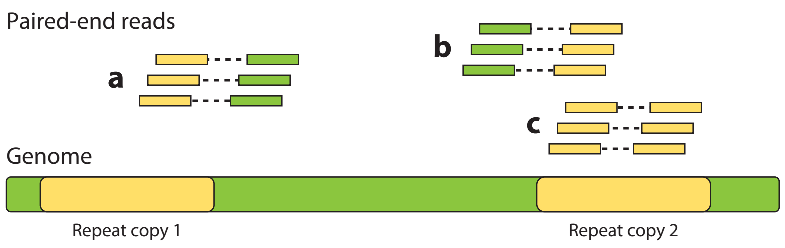


Figure 9

An illustration of how paired-end reads can assist in finding the true point of origin of reads originating inside repetitive elements of the genome. Each paired-end read is illustrated as a pair of rectangles joined by a dotted line that represents the unsequenced portion of the fragment. The pairs labeled *a* and *b* are those where one end falls within the repeat (*yellow region*) and the other end falls outside the repeat (*green region*). In these cases, the end that falls outside the repeat assists in identifying the point of origin for the end that falls inside the repeat. The pairs labeled *c* are those where both ends fall within the repeat. In these cases, the read's point of origin is still ambiguous.

Repeats

Repetitive DNA elements (repeats) are prevalent in genomes, especially those of higher eukaryotes. Repeats make up approximately 50% of the human genome (31) and more than 80% of the maize genome (57). They can be categorized as interspersed, where similar DNA sequences are spread throughout the genome, or tandem, where similar sequences are adjacent (63). Some interspersed repeats are long segmental duplications, but most are relatively short transposons and retrotransposons. Although repeats are sometimes referred to as "junk," they are involved in intrinsic biological processes, including genome expansion, speciation, and epigenetic regulation (16). Some are still actively expressed and duplicated, including some in the human genome (64, 69).

Using short DNA sequences such as NGS reads to study repetitive genomes raises methodological and interpretation questions (63). For instance, if a read originates from a repeat family, how can a read aligner identify the exact instance from which the read came? Sometimes this simply is not possible, especially when copies of the repeat are exactly the same. Because this is a possibility, it is useful for the read aligner to report a degree of confidence in its alignment.

Modern read aligners attempt to measure this confidence by considering all of the alignments discovered in the process of aligning a read. For instance, if a read aligns equally well to several instances of a repeat, then the aligner might report low confidence. On the other hand, if the read aligns perfectly to one locus and very poorly to a few other loci, then the aligner might report high confidence. Scientists and downstream tools (e.g., variant callers) should be careful to take this confidence into account when weighing evidence derived from the alignment.

How exactly does the aligner assess its confidence? Previous studies have shown how information about all alignments found during the search (not just the highest scoring) can be used to estimate the probability that the highest-scoring alignment is correct (36). By correct, we mean that the read was placed in its true point of origin with respect to the reference. Call this probability p_{cor}, and let $Q = -10\log_{10}(1 - p_{\mathrm{cor}})$. Q is the mapping quality, and it is standard [e.g., in the Sequence Alignment/Map (SAM) format (35)] to report Q with each alignment. Q is on the Phred scale: $Q = 10$ conveys that the aligner estimates that there is a 1 in 10 chance that the alignment is incorrect, $Q = 20$ conveys a 1 in 100 chance, $Q = 30$ conveys a 1 in 1,000 chance, and so on.

Note that mapping quality and alignment score are distinct measures. A high alignment score implies a large degree of similarity (i.e., few mismatches and gaps) between the read and the reference, but does not imply high mapping quality. For instance, consider a read that aligns perfectly to the genome in two distinct loci. The alignment score is high, but intuitively, there is approximately a 50% chance of choosing the incorrect alignment ($Q \leq 3$). Any other measure that does not explicitly consider the repeat content of the genome [e.g., BLAST *E* values (29)] will also be a poor proxy for mapping quality.

Sequence Differences from the Reference

A key point when interpreting read alignments is that the sequence of the subject genome is not identical to the sequence of the reference genome. The subject and reference genomes may be extremely similar (e.g., as noted above, the genomes of two unrelated humans are approximately 99.8% similar by sequence), but they are not identical. Furthermore, differences that exist between genomes are often distributed unevenly along the genome's length. This is true for the human genome, both for variants generally (65) and for indels in particular (45), and can lead to interpretability issues. For example, say that one has aligned a collection of sequencing reads and finds that, whereas most of the reference is covered nearly uniformly at a high average depth, some regions have little or no coverage. This could be evidence of a deletion in the subject genome, or could have arisen because the poorly covered regions contained so many sequence differences from the reference that the aligner failed to find the alignments that truly belonged in the region.

Others have noted and summarized these issues (12, 40, 51, 71), and a few approaches have been proposed to resolve them. One involves substituting the reference genome for another reference that more closely matches the subject genome. For example, if the subject is human and the ethnicity is known, a version of the reference genome from that ethnicity could be used. The tailored reference genome would differ from the reference genome such that common variants in the relevant population are used. Previous studies have assembled such references for human ethnicities (13) and for accessions of *Arabidopsis thaliana* (19). A related idea is to use variant information inferred from read alignments to modify the reference genome to more closely resemble the subject genome (20). The alignment process can then be restarted using the new, tailored reference genome. This process can also be iterated until there are no more changes to be made to the reference.

Another way to reduce the effect of differences between subject and reference genomes is to encode information about small-scale genetic variants in the reference sequence itself. For example, say a particular position in the genome is known to vary across individuals, and the alleles found there are either C or T. One could replace the character at that position with Y, the International Union of Pure and Applied Chemistry (IUPAC) code representing "either C or T." Read aligners that respect this convention are sometimes called single-nucleotide polymorphism (SNP) aware or SNP tolerant (24, 71), and they avoid incurring a penalty when, for example, a C or T aligns opposite a Y in the reference genome. This removes alignment score penalties associated with a nonreference allele, which in turn increases the fraction of reads for which the aligner finds an alignment (24, 71). Such tools require that the user specify an annotation file describing the SNP variants to be included in the reference genome.

A still more general approach is to avoid representing the reference as a string at all and instead to represent it as a graph. A graph is a diagram that allows any kind of genetic variant, large or small, to be represented as a path through the space of possible ways of gluing sequences together to form a genome. Past studies have considered the question of how to construct and represent genome graphs in a way that is both memory efficient and fast to query (18, 41). Recent software tools make it easy to construct such graphs from genome sequences (43). Other tools allow users to both build the graph and align reads to it (58).

Ploidy and Allelic Alignment Bias

Say that one is analyzing sequencing reads from a human genome, which is diploid, and considering a position on the subject genome where there is a heterozygous SNP. Typically, one of the alleles of the SNP will match the allele in the reference genome (the reference allele), whereas the other will not. When aligning a read containing the reference allele, the read will match the reference genome at that position, but when aligning a read containing the nonreference allele, the read will mismatch at that position, reducing the alignment score. In some situations, the aligner might fail to align reads containing many nonreference alleles, which in turn leads to an underrepresentation of reads from the haplotype with more nonreference alleles. For DNA sequencing data, this tends to drive the allelic balance away from the expected 1-to-1 ratio. Departure from the 1-to-1 ratio must therefore be expected by downstream tools, such as variant callers.

Some solutions to this problem have been proposed specifically in the context of RNA sequencing and the problem of detecting allele-specific expression (12, 21). One solution is to mutate reference positions where heterozygous SNPs might occur in a third allele that is neither the reference allele nor the alternate allele (12). This alleviates the bias by ensuring that reference and alternate alleles are penalized equally. Another solution is to use SNP-aware alignment. If the annotation provided to the SNP-aware aligner contains both alleles for a heterozygous position, then the aligner will not penalize alignments that overlap with the nonreference allele at that position, also alleviating the bias (55). The efficacy of this method depends on what proportion of the relevant alleles are present in the annotation. Another proposed method builds a custom, phased diploid reference genome for a human individual under study based on extensive genotype information for that individual and his or her ancestors (52). This reduces the bias in question but requires more genotype information than is typically available.

MANY GENOMES

All the difficulties and problems mentioned above are multiplied if one is investigating many reference genomes at once in the context of metagenomic sequencing data. Read alignment also plays a key role here. Rather than attempting to isolate a particular species, metagenomic sequencing reads are derived from all of the various species present in a sample. The sample could be from any environment expected to contain life, e.g., the human gut (26), seawater (54), or soil (10).

Given a collection of reads from a metagenomics experiment, one might want to determine what organisms are present and in what abundance. For bacteria, one could begin by aligning the reads to all known bacterial reference genomes. The best alignment of the read over all of the reference genomes would be the best guess about which genome it originated from. The number of reads aligning to each genome (perhaps normalized by genome length) is a measure of each genome's abundance.

A problem unique to this setting is that reads often align equally well to many genomes. For a given bacterial species, there might be many strains that have been sequenced and for which a reference genome is available. For instance, the RefSeq database contains finished reference genomes for more than 50 strains of *Helicobacter pylori*. These strains tend to have very similar genome sequences, so a read derived from one strain will tend to align well to many strains. Similar ambiguity could exist at higher levels of the taxonomic tree; for example, a read might align equally well to genomes of multiple species, or multiple genera, or multiple families. For this reason, metagenomic analyses take the taxonomic tree into account when drawing conclusions from read alignments. If a read aligns unambiguously to a particular genus, then this is strong evidence that the read originated from that genus. But if the same read aligns ambiguously to many species within the genus, there is not strong evidence for any particular species.

Another concern is that a read could be derived from an organism whose genome has not been sequenced or is otherwise not represented in the collection of reference genomes. If this organism is closely related to one of these reference genomes, then one might still be able to salvage information at higher levels of the taxonomic tree, albeit at the risk of drawing spurious conclusions at lower levels. If this organism is not closely related to any of the reference genomes, then the investigator might learn nothing from that read.

Many software tools for analyzing metagenomic data sets are based on read alignment (6, 27). Some tools additionally use marker sequences, which are sequences derived from reference genomes that are unique to a particular taxonomic category, to reduce the total size of the reference sequences being searched against (39, 59). Other tools avoid solving the approximate matching problem by instead searching the reference genomes for exact q-gram matches between reads and reference sequences (3, 70). These so-called alignment-free solutions are simpler and often faster than alignment-based solutions, but they are sensitive to the choice of q.

CONCLUSION AND FUTURE DIRECTIONS

The last decade has brought exciting new technologies for genomic sequencing, which in turn has triggered a plethora of new developments for the well-known and well-understood problem of approximate string matching. New algorithms and implementations have brought alignment programs that are orders of magnitude faster than those from ten years ago.

We predict that, in the future, most improvements for the approximate read alignment problem will come from the use of hardware accelerators such as general-purpose graphics processors or coprocessors such as the Intel Xeon Phi. With respect to the data, it will be interesting to see whether the community will come up with new ideas about how to solve the ambiguity problem, especially for highly repetitive genomes. It stands to reason that postprocessing of all ambiguous matching positions is needed.

DISCLOSURE STATEMENT

D.J.E. was employed by Illumina from 2007 to 2012.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation grant IIS 1349906 and a Sloan Research Fellowship (B.L.) and by the German Federal Ministry of Education and Research (D.W.).

LITERATURE CITED

1. Abouelhoda MI, Kurtz S, Ohlebusch E. 2004. Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms* 2:53–86
2. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. 1990. Basic local alignment search tool. *J. Mol. Biol.* 215:403–10
3. Ames SK, Hysom DA, Gardner SN, Lloyd GS, Gokhale MB, Allen JE. 2013. Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics* 29:2253–60
4. Baeza-Yates RA, Navarro G. 1999. Faster approximate string matching. *Algorithmica* 23:127–58
5. Bradnam KR, Fass JN, Alexandrov A, Baranay P, Bechner M, et al. 2013. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *Gigascience* 2:10
6. Brady A, Salzberg SL. 2009. Phymm and Phymmbl: metagenomic phylogenetic classification with interpolated Markov models. *Nat. Methods* 6:673–76

7. Brandon R, Myers GW, Sutton GG, Delcher AL, Dew IM, et al. 2000. A whole-genome assembly of *Drosophila*. *Science* 287:2196–204
8. Burkhardt S, Crauser A, Ferragina P, Lenhof H-P, Rivals E, Vingron M. 1999. *q*-gram based database searching using a suffix array (Quasar). In *RECOMB '99: Proceedings of the Third Annual International Conference on Computational Molecular Biology*, pp. 77–83. New York: Am. Chem. Soc.
9. Burrows M, Wheeler DJ. 1994. *A block-sorting lossless data compression algorithm*. Res. Rep. 124, Syst. Res. Cent., Palo Alto, CA
10. Daniel R. 2005. The metagenomics of soil. *Nat. Rev. Microbiol.* 3:470–78
11. David M, Dzamba M, Lister D, Ilie L, Brudno M. 2011. SHRiMP2: sensitive yet practical short read mapping. *Bioinformatics* 27:1011–12
12. Degner JF, Marioni JC, Pai AA, Pickrell JK, Nkadori E, et al. 2009. Effect of read-mapping biases on detecting allele-specific expression from RNA-sequencing data. *Bioinformatics* 25:3207–12
13. Dewey FE, Chen R, Cordero SP, Ormond KE, Caleshu C, et al. 2011. Phased whole-genome genetic risk in a family quartet using a major allele reference sequence. *PLOS Genet.* 7:e1002280
14. Dohm JC, Lottaz C, Borodina T, Himmelbauer H. 2008. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Res.* 36:e105
15. Farrar M. 2007. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics* 23:156–61
16. Fedoroff NV. 2012. Transposable elements, epigenetics, and genome evolution. *Science* 338:758–67
17. Ferragina P, Manzini G. 2000. Opportunistic data structures with applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 12–14 November 2000, Redondo Beach, California*, pp. 390–98. Los Alamitos, CA: IEEE Comput. Soc.
18. Ferragina P, Mishra B. 2014. Algorithms in stringomics (I): pattern-matching against "stringomes." bioRxiv. doi: 10.1101/001669
19. Gan X, Stegle O, Behr J, Steffen JG, Drewe P, et al. 2011. Multiple reference genomes and transcriptomes for *Arabidopsis thaliana*. *Nature* 477:419–23
20. Ghanayim A, Geiger D. 2013. *Iterative referencing for improving the interpretation of DNA sequence data*. Tech. Rep., Technion, Haifa, Isr.
21. Gilad Y, Pritchard JK, Thornton K. 2009. Characterizing natural variation using next-generation sequencing technologies. *Trends Genet.* 25:463–71
22. Glenn TC. 2011. Field guide to next-generation DNA sequencers. *Mol. Ecol. Resour.* 11:759–69
23. Hach F, Hormozdiari F, Alkan C, Hormozdiari F, Birol I, et al. 2010. mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nat. Methods* 7:576–77
24. Hach F, Sarrafi I, Hormozdiari F, Alkan C, Eichler EE, Sahinalp SC. 2014. mrsFAST-Ultra: a compact, SNP-aware mapper for high performance sequencing applications. *Nucleic Acids Res.* 42:W494–500
25. Hoffmann S, Otto C, Kurtz S, Sharma CM, Khaitovich P, et al. 2009. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLOS Comput. Biol.* 5:e1000502
26. Hum. Microbiome Proj. Consort. 2012. Structure, function and diversity of the healthy human microbiome. *Nature* 486:207–14
27. Huson DH, Auch AF, Qi J, Schuster SC. 2007. MEGAN analysis of metagenomic data. *Genome Res.* 17:377–86
28. Jiang H, Wong WH. 2008. SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics* 24:2395–96
29. Karlin S, Altschul SF. 1990. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *PNAS* 87:2264–68
30. Kiełbasa SM, Wan R, Sato K, Horton P, Frith MC. 2011. Adaptive seeds tame genomic sequence comparison. *Genome Res.* 21:487–93
31. Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, et al. 2001. Initial sequencing and analysis of the human genome. *Nature* 409:860–921
32. Langmead B, Salzberg S. 2012. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* 9:357–59
33. Langmead B, Trapnell C, Pop M, Salzberg SL. 2009. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.* 10:R25

34. Li H, Durbin R. 2009. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 25:1754–60
35. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, et al. 2009. The sequence alignment/map format and SAMtools. *Bioinformatics* 25:2078–79
36. Li H, Ruan J, Durbin R. 2008. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.* 18:1851–58
37. Li R, Li Y, Kristiansen K, Wang J. 2008. SOAP: short oligonucleotide alignment program. *Bioinformatics* 24:713–14
38. Li R, Yu C, Li Y, Lam T-W, Yiu S-M, et al. 2009. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics* 25:1966–67
39. Liu B, Gibbons T, Ghodsi M, Treangen T, Pop M. 2011. Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. *BMC Genomics* 12(Suppl. 2):S4
40. Lunter G, Goodson M. 2011. Stampy: a statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Res.* 21:936–39
41. Mäkinen V, Navarro G, Sirén J, Välimäki N. 2009. Storage and retrieval of individual genomes. In *Research in Computational Molecular Biology: 13th Annual International Conference*, *RECOMB 2009*, *Tucson*, *AZ*, *USA*, *May 18–21*, *2009*, *Proceedings*, ed. S Batzoglou, pp. 121–37. Lect. Notes Bioinform. 5541. Berlin: Springer
42. Manber U, Myers EW. 1990. Suffix arrays: a new method for on-line string searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 319–27. Philadelphia: Soc. Ind. Appl. Math.
43. Marcus S, Lee H, Schatz MC. 2014. SplitMEM: a graphical algorithm for pan-genome analysis with suffix skips. *Bioinformatics* 30:3476–83
44. Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, et al. 2005. Genome sequencing in microfabricated high-density picolitre reactors. *Nature* 437:376–80
45. Montgomery SB, Goode DL, Kvikstad E, Albers CA, Zhang ZD, et al. 2013. The origin, evolution, and functional impact of short insertion–deletion variants identified in 179 human genomes. *Genome Res.* 23:749–61
46. Myers EW. 1986. An O(ND) difference algorithm and its variations. *Algorithmica* 1:251–66
47. Myers EW. 1999. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *J. ACM* 46:395–415
48. Nagarajan N, Pop M. 2013. Sequence assembly demystified. *Nat. Rev. Genet.* 14:157–67
49. Pagh R, Rodler FF. 2001. Cuckoo hashing. In *Algorithms: ESA 2001*, ed. FM auf der Heide, pp. 121–33. Lect. Notes Comput. Sci. 2161. Berlin: Springer
50. Rasmussen KR, Stoye J, Myers EW. 2006. Efficient *q*-gram filters for finding all ε-matches over a given length. *J. Comput. Biol*. 13:296–308
51. Rosenfeld JA, Mason CE, Smith TM. 2012. Limitations of the human reference genome for personalized genomics. *PLOS ONE* 7:e40294
52. Rozowsky J, Abyzov A, Wang J, Alves P, Raha D, et al. 2011. AlleleSeq: analysis of allele-specific expression and binding in a network framework. *Mol. Syst. Biol.* 7:522
53. Rumble SM, Lacroute P, Dalca AV, Fiume M, Sidow A, Brudno M. 2009. SHRiMP: accurate mapping of short color-space reads. *PLOS Comput. Biol.* 5:e1000386
54. Rusch DB, Halpern AL, Sutton G, Heidelberg KB, Williamson S, et al. 2007. The *Sorcerer II* Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific. *PLOS Biol.* 5:e77
55. Satya RV, Zavaljevski N, Reifman J. 2012. A new strategy to reduce allelic bias in RNA-Seq readmapping. *Nucleic Acids Res.* 40:e127
56. Schatz MC, Delcher AL, Salzberg SL. 2010. Assembly of large genomes using second-generation sequencing. *Genome Res.* 20:1165–73
57. Schnable PS, Ware D, Fulton RS, Stein JC, Wei F, et al. 2009. The B73 maize genome: complexity, diversity, and dynamics. *Science* 326:1112–15
58. Schneeberger K, Hagmann J, Ossowski S, Warthmann N, Gesing S, et al. 2009. Simultaneous alignment of short reads against multiple genomes. *Genome Biol.* 10:R98
59. Segata N, Waldron L, Ballarini A, Narasimhan V, Jousson O, Huttenhower C. 2012. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nat. Methods* 9:811–14

60. Siragusa E, Weese D, Reinert K. 2013. Fast and accurate read mapping with approximate seeds and multiple backtracking. *Nucleic Acids Res.* 41:e78
61. Smith TF, Waterman MS. 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147:195–97
62. Stein L. 2010. The case for cloud computing in genome informatics. *Genome Biol.* 11:207
63. Treangen TJ, Salzberg SL. 2011. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nat. Rev. Genet.* 13:36–46
64. Tyekucheva S, Yolken RH, McCombie WR, Parla J, Kramer M, et al. 2011. Establishing the baseline level of repetitive element expression in the human cortex. *BMC Genomics* 12:495
65. Venter JC, Adams MD, Myers EW, Li PW, Mural RJ, et al. 2001. The sequence of the human genome. *Science* 291:1304–51
66. Weese D, Emde A, Rausch T, Döring A, Reinert K. 2009. RazerS—fast read mapping with sensitivity control. *Genome Res.* 19:1646–54
67. Weese D, Holtgrewe M, Reinert K. 2012. RazerS 3: faster, fully sensitive read mapping. *Bioinformatics* 28:2592–99
68. Weiner P. 1973. Linear pattern matching algorithms. In *Proceedings of the 14th Annual Symposium on Switching and Automata Theory*, pp. 1–11. Northridge, CA: IEEE Comput. Soc.
69. Witherspoon D, Xing J, Zhang Y, Watkins WS, Batzer M, Jorde L. 2010. Mobile element scanning (ME-Scan) by targeted high-throughput sequencing. *BMC Genomics* 11:410
70. Wood D, Salzberg S. 2014. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* 15:R46
71. Wu TD, Nacu S. 2010. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics* 26:873–81