Sense from sequence reads: methods for alignment and assembly

Paul Flicek & Ewan Birney

The most important first step in understanding next-generation sequencing data is the initial alignment or assembly that determines whether an experiment has succeeded and provides a first glimpse into the results. In parallel with the growth of new sequencing technologies, several algorithms that align or assemble the large data output of today's sequencing machines have been developed. We discuss the current algorithmic approaches and future directions of these fundamental tools and provide specific examples for some commonly used tools.

The advent of ultra-high-throughput sequencing technology has captured the imagination of the biological sciences, and with good reason. Ten years ago, on 23 November 1999, the publicly funded human genome project held a massive, worldwide celebration to mark the completion of 1 billion base pairs (bp), one-third of the way to the full sequence of the human genome (http://www.genome.gov/10002105/). The amount of sequence was so incredible at the time that the celebration featured senators and US cabinet officials. Commemorative T-shirts marking the occasion were distributed. Today, sequencing 1 billion bp is the work of hours in any lab equipped with an Illumina GAII or ABI SOLiD 'second generation' sequencing machine and the work of minutes in large-scale sequencing centers. These large centers may have 40 or more such machines sitting in the middle of a massive production pipeline that requires a substantial wet-lab work flow to feed the sequencing machines and almost unimaginable computational support to make any sense of the data coming from the sequencers. All indications suggest we have only scratched the surface of the potential for ubiquitous DNA sequencing technology to change the way experiments are conducted and biology is understood.

One thing that has not changed in the last 10 years is that the individual outputs of the sequence machines are essentially worthless by themselves. The individual 'reads' (named so from the days when the sequence of a given DNA molecule was determined by a pair of human eyes looking down an autoradiograph of a gel that had a separate lane for each base) range from approximately 800 bp, using the older technology used by the human genome project, to approximately 30 bp for the introductory versions of the second-generation sequencing machines so popular today. Current output ranges from 50 to 400 bp, depending on the technology and the specific biological application. Although uninformative by themselves, once analyzed collectively DNA sequencing reads have tremendous versatility, and the existing applications of next-generation sequencing are extensive. Fundamental to creating biological understanding from the increasing piles of sequence data is the development of analysis algorithms able to assess the success of the experiments and synthesize the data into manageable and understandable pieces.

We will focus on two of the most fundamental computational analyses in the context of sequence analysis: alignment and assembly. When a reference genome assembly exists (for example, for human or mouse), alignment remains the first and most fundamental analysis once the DNA sequence has been produced. The results of the alignment have the ability to quickly determine whether the sequencing experiment has succeeded, whether the correct sample was sequenced, and whether the biological experiment and DNA preparation succeeded. For organisms without a sequenced reference genome, assembly is almost always essential for analysis. However, in order to develop algorithms to accurately assemble new genomes, the existence of already assembled reference genomes in other species is

European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, UK. Correspondence should be addressed to P.F. (flicek@ebi.ac.uk).

PUBLISHED ONLINE 15 OCTOBER 2009; CORRECTED ONLINE 6 MAY 2010; DOI:10.1038/NMETH.1376

critical because they allow newly developed algorithms to be benchmarked against the known solutions, which then gives confidence to the assembly results for species without a reference genome.

The choice of alignment or assembly algorithm is strongly influenced by both the experiment in question and the details of the sequencing technology used. The performance characteristics of the sequencing machines are changing rapidly, and any delineation of performance characteristics such as machine capacity, run time or read length and its relationship to error profile will quickly be outdated. In this review, thus, we will instead describe the types of data that are likely to be generated for specific experimental applications, with some confidence that (i) both sequence capacity and quality will continue to increase for all platforms and that it will be possible to generate high-quality sequence of various read lengths and (ii) an optimal quality versus cost tradeoff will appear for given experimental applications. For example, *de novo* sequencing for large genomes will benefit from longer reads than are now available, high coverage, and paired-end reads with multiple, well-chosen insert sizes. Similarly, a resequencing application done in the presence of a reference genome assembly requires reads that can be accurately mapped in such a way that both nucleotide and structural variation can be reliably assessed; it may be possible to do this with paired reads on the order of 100 bp in length, although longer reads may be beneficial[1]. At the same time, a chromatin immunoprecipitation (ChIP)-sequencing experiment, for mapping transcription factor binding sites or the location of modified histones, isolates relatively short fragments of DNA sequence and has little need for reads longer than 50–75 bp and apparently limited benefit from paired-end reads[2]. Other applications, such as DNase-seq, are limited to only 20 bp of informative sequence and so are unaffected by both longer read lengths and paired-end sequences[3]. Transcriptomics (for example, RNA-seq) experiments will also have optimal experimental designs[2].

There are two fundamental considerations when designing alignment and assembly algorithms for all second-generation sequencing data, beyond the obvious consideration that the reads are shorter than with gel-capillary technology. First, the amount of data produced is orders of magnitude greater than that generated by earlier techniques, so any algorithm must be optimized for speed and memory usage. Second, the techniques produce data with different error profiles than the previous-generation technology, which must be addressed at the algorithmic level to obtain the maximum information from the sequencing data. Gel capillary reads normally had low quality base calls at the start and the end of the read, with high-quality data in the central region. Base-calling algorithms provided information about where a given read should be 'quality clipped' for users who wanted only the best part of the read. The different error characteristics with second-generation technology include, for example, the tendency of Roche 454 reads to have insertion or deletion errors during homopolymer runs[4] and the increasing likelihood of sequence errors toward the end of the read for ABI SOLiD and Illumina/Solexa technology. As a further consideration, SOLiD data are produced in 'color space' rather than the 'base space' used by the other sequencing technologies, meaning that the output of the SOLiD machine is a series of colors representing two nucleotides (represented by the numbers 0–3) rather than a series of bases (represented by A, C, G and T). The SOLiD color-space code is degenerate, such that only four colors are used to represent the 16 possible transitions (for example, the dinucleotides AA, CC, GG and TT are all represented by the same color). Hence, some sequence errors are correctable, assuming that the analysis tools explicitly consider this aspect of the data[5]. The Illumina technology uses other techniques to remove likely errors earlier in the processing pipeline (that is, before the alignment or assembly process), including 'purity filtering' reads appearing to come from more than one DNA molecule[6].

Alignment

Alignment itself is the process of determining the most likely source within the genome sequence for the observed DNA sequencing read, given the knowledge of which species the sequence has come from. Sequencing reads may also be aligned to other genomes, assuming the evolutionary distance between the genomes is appropriate. The most widely used alignment programs for second-generation sequence data have been explicitly designed (or modified) for the purpose of aligning this data. Unlike earlier-generation sequence alignment programs such as BLAST, which were designed in an environment that required alignments of protein sequences and searching though large databases to find homologous sequences, today's short-read alignment programs are generally used for the alignment of DNA sequence from the species of interest to the reference genome assembly of that species. This difference, although it initially seems subtle, has several consequences to the final algorithm design and implementation, which include letting assumptions about the number of expected mismatches be driven by the species polymorphism rate and the technology error rate rather than by considerations of evolutionary substitutions.

In general, these assumptions allow for much faster processing, as few low-quality alignments are either expected or scored. Given the massive data volumes produced by the present sequencing machines, this has also allowed alignments to be calculated without a correspondingly massive increase in computer hardware requirements. As sequence capacity grows, algorithmic speed may become a more important bottleneck.

Although there is a large and ever growing number of implementations for short-read sequence alignment, the number of fundamental technologies used is much smaller. We will focus on two such techniques, give a general overview of the methods describing some of the advantages and disadvantages, and provide examples from some of the most commonly used implementations of the method. The methods covered are (i) hash table–based implementations, in which the hash may be created using either the reference genome or the set of sequencing reads, and (ii) Burrows Wheeler transform (BWT)-based methods, which first create an efficient index of the reference genome assembly in a way that facilitates rapid searching in a low-memory footprint. Both of the above methods can be applied to color-space (SOLiD) reads or base-space (Illumina, 454) reads, although this capacity must be designed into the alignment program.

Alignment programs normally follow a multistep procedure to accurately map sequence. Using heuristic techniques in the first step, efforts are made to quickly identify a small set of places in the reference sequence where the location of the best mapping is most likely to be found. Once the smaller subset of possible mapping locations has been identified, slower and more accurate alignment algorithms such as Smith-Waterman are run on the limited subset (see ref. 7 for review). Running these accurate alignment algorithms as a full search of all possible places where the sequence may map is computationally infeasible. This section of the review will concentrate on algorithms for the first step that are particularly appropriate for short-read data and only briefly mention the algorithms used in the second step (although these can be important for the fine-scale results). Additionally, all of the programs implement a 'mapping policy' that governs key performance aspects of the specific implementation. Regardless of the underlying algorithmic approach, a general rule is that there is tradeoff between speed and sensitivity. That is, a procedure that can map reads with guaranteed high accuracy, especially in the presence of errors and sequence polymorphism, will take longer than a procedure that applies heuristics to limit the problem in one way or another.

Hash-based alignment methods. The first wave of alignment programs specifically designed for short-read alignment from next-generation sequencing machines was based on a hash-table data structure to index and scan the sequence data. 'Hash table' refers to a common data structure that is able to index complex and nonsequential data in a way that facilitates rapid searching (**Fig. 1**). This is especially appropriate for DNA sequencing reads, which are extremely unlikely to contain every possible combination of nucleotides and very likely to contain duplicates. Examples of tools using this approach include MAQ[8], SOAP[9] and Illumina's own unpublished ELAND algorithm. These have recently been joined by several other tools, including SHRiMP[10], ZOOM[11], BFAST (http://genome.ucla.edu/bfast/) and MOSAIK (http://bioinformatics.bc.edu/marthlab/Mosaik/).

Hash-based algorithms build their hash table either on the set of input reads or on the reference genome. They then use the reference genome to scan the hash table of input reads (in the first case) or use the set of input reads to scan the hash table of the reference genome (in the second). There are advantages and disadvantages to each method. For example, hash tables of the reference genome have a constant memory requirement for a given parameter set regardless of the size of the input set of reads, which may be large, depending on the size and complexity of the reference genome. Hash tables based on the set of input reads typically have smaller and variable memory requirements based on the number and diversity of the input read set but may use more processing time to scan the entire reference genome when there are relatively few reads in the input set. Of the algorithms mentioned above, MAQ, ELAND, ZOOM and SHRiMP build a hash table of the input read sequences, whereas SOAP, BFAST and MOSAIK hash the reference genome assembly.

With either hashing methodology, the algorithms typically implement the hash table in the form of 'spaced seeds', which are regions of the sequence required to have a specific pattern of matches and mismatches. Spaced seeds were popularized for sequence alignment by the PatternHunter program[12]. A seed is of the form 110011, where 1 represents a position of the sequence that is required to match and the number of 1s is designated the 'weight' of the seed. For example, from the first 28 bp of the read, the published MAQ program builds six hash tables corresponding to seeds of length 8 and weight 4 and then scans the reference genome against these hash tables[8]. This technique ensures that all hits with two mismatches can be found, and more than half of those with three mismatches. Twenty hash tables would be required for MAQ to guarantee that all reads with three mismatches could be found. ZOOM uses manually constructed spaced seeds of weight 14 to enable the detection of up to four mismatches in 50-bp reads[12] and SHRiMP uses a q-gram approach[13], which requires that multiple spaced seeds per read

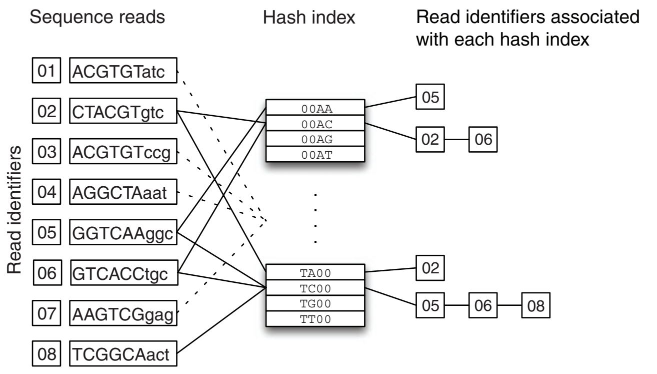

Figure 1 | Schematic of a hash table–based alignment strategy. Sequence reads with associated read identifiers are shown, with the regions that will be used for seed selection in capital letters and matched seeds of 0011 and 1100. Given read identifiers are associated with the seeds using a hash function (for example, a unique integer representation of each seed). Once such a hash table has been built for either the input read set or the reference genome, the corresponding data can be scanned with the same hash function, resulting in a much smaller subset of reads to more exactly align at each location in the genome.

match if a region is to be considered a possible alignment location[10]. In its recommended use, MOSAIK hashes all positions in the reference genome and uses a 'jump database' to efficiently locate information in the hash table and thus reduces memory requirements by approximately two-thirds over a naive implementation.

After the alignment seeds have been used in the hash table creation and the reads have been associated with the region of the genome where they are most likely to align, a specialized and accurate alignment algorithm is used to determine the exact placement of the sequence reads on the reference genome. Such algorithms include both gapped and ungapped versions of Smith-Waterman that take advantage of the quality values of the sequenced bases.

Burrows-Wheeler transform methods. Over the past year, a new generation of short-read alignment programs including BOWTIE[14], BWA[15] and SOAP2 (ref. 16) have been developed that are based on the Burrows-Wheeler transform (BWT)[17]. These methods typically use the FM index data structure, proposed by Ferragina and Manzini, who introduced the concept that a suffix array is much more efficient if it is created from the BWT sequence, rather than from the original sequence[18]. The FM index retains the suffix array's ability for rapid subsequence search and, for mammalian genomes, is often the same size or smaller than the input genome size[19]. For example, the final index for the human genome used by both BWA and BOWTIE is approximately 2.3 GB in size[14,15], whereas SOAP2 uses a different routine resulting in a final index that requires 5.4 GB (ref. 16).

Creating the underlying data structure requires two steps. In the first step, the sequence order of the reference genome is modified using the BWT, a reversible process (that is, the original genome sequence can easily be reconstructed) that reorders the genome such that sequences that exist multiple times appear together in the data structure (**Fig. 2**). Next, the final index is created; it is then used for rapid read placement on the genome. The creation of the final index may be a memory-intensive step, although methods exist to create the index in relatively little memory at the cost of more processing time[20]. The BWT has been commonly used in data compression; thus, the FM index structure has been referred to as a compressed suffix array[18].

BWT implementations are much faster than their hash-based counterparts at the same sensitivity level and can be several times faster still at slightly reduced sensitivity levels and for single-ended reads. This last case is likely to be most appropriate for mapping tag sequences from ChIP-seq or similar applications. Another advantage to the BWT-based methods is the ability to store the complete reference genome index on disk and load it completely into memory on almost all standard bioinformatics computing clusters[21].

BOWTIE's reported 30-fold speed increase over hash-based MAQ is an example of the speed increases for single-ended reads, although this increase is at the cost of a small loss of alignment sensitivity[13]. With the ability to exploit paired-end reads and for sensitivity similar to the earlier, hash-based methods, the speed increase for any of the current BWT-based programs will generally be tenfold[14–16].

The limitations of BWT-based methods are another example of tradeoff between speed and sensitivity. For example, BWA is only able to find alignments within a certain 'edit distance' of the sequence in reference genome, which is a function of the read length[15]. Edit distance is, formally, the number of operations required to transform one sequence into another, which in the case of sequence alignment is most commonly gaps or mismatches. This effectively limits the combined number of mismatches or gaps in the read that can be aligned (for 100-bp reads, BWA allows 5 'edits'; less for shorter reads and more for longer reads). As sequencing becomes increasingly accurate, this limitation is likely to become less important for species with relatively low polymorphism rates, such as human where nearly all reads will align within the edit distance.

As in the hash-based methods, once the reads have been associated with the region of the genome where they are most likely to align, more sensitive algorithms can be used for the final alignment result.

Assembly

A fundamental goal of DNA sequencing has been to generate large, continuous regions of DNA sequence. The desired DNA sequences are nearly always far longer than the individual sequencing read lengths, so some sort of sampling approach is required. The dominant approach has been to randomly fragment a DNA sequence, sequence these fragments—described as 'shotgun sequencing'—and then reconstruct the original DNA sequence computationally, described as 'assembly'. This was originally designed for cosmid or other clone resources[22,23] but also became feasible for genomes, first of bacteria and then of large, complex eukaryotes.

With the old-technology read lengths (around 800 bp), all assembly algorithms worked as some variant of using overlaps between reads and then a resolution of these overlaps into a colinear solution. With the far shorter reads and far higher coverages generated by these new technologies, not only was this 'read-centric' method computationally unfeasible, but it was seemingly impossible to find heuristics to resolve the large number of overlaps. However, pioneering work by Pevzner and colleagues[24] in the late 1980s and Idury and Waterman[25] in the mid-1990s had already introduced a different framework for handling assemblies, even with the older, long-read technology. The new framework was based on a graph of very small, fixed-length subsequences (abbreviated as *k*-mers, where *k* is 19 or higher) in a de Bruijn graph data structure, which was originally developed for combinatorial mathematics. For applications in DNA sequence assembly, the de Bruijn graph has a node for every *k*-mer observed in the sequence set and an edge between nodes if these two *k*-mers are observed adjacently in a read. Such edges are therefore associated with the single-step base difference of moving the fixed *k*-mer window along by one position. Although seemingly similar to the read-overlap graph used by traditional assembly programs, which often use *k*-mer content to calculate the overlaps, the de Bruijn graph formulation has properties that differ in important ways. The first is that a read will be split across its component nodes. The second is how this structure handles repeats: a repeat will be a series of adjacent *k*-mers which many reads pass through. On the edges of the repeat, the graph will diverge into the unique regions of the genome. A final aspect of this graph is that it can be constructed in an amount of computational time that scales linearly with the number of reads (rather than in quadratic time, as is needed for the naive, all-against-all implementation of the overlap graph).

Pevzner and Tang used this data structure to find solutions based on a graph traversal method—namely, an eulerian tour of the graph[26], which visits each node of the graph exactly once. This graph traversal method was not particularly successful for making practical assemblies, but the de Bruijn graph framework is ideal for handling high-coverage, short sequencing reads and has several useful properties—for example, being able to easily compute the theoretical maximum continuity for a particular sequence

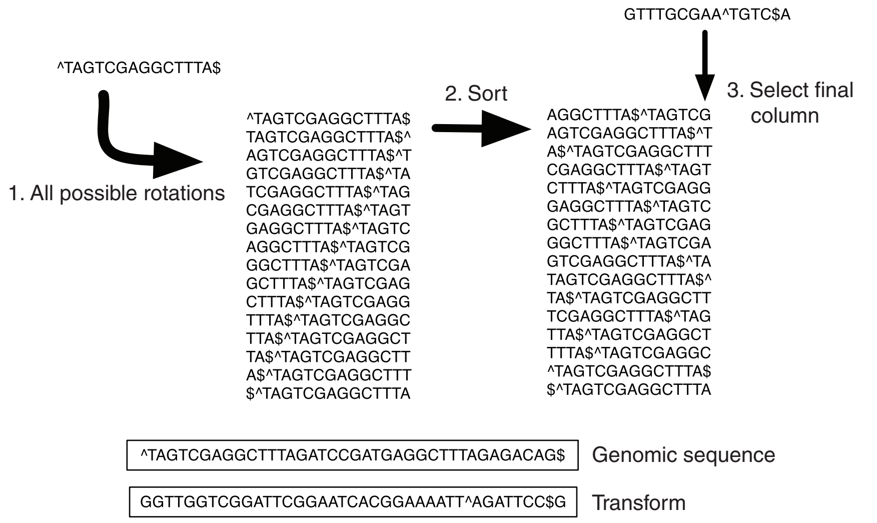

Figure 2 | The Burrows-Wheeler transform for genomic sequence data. To create a BWT of a 14-mer genomic sequence, one first notes the start and end points of the sequence and then constructs all rotations of the given sequence by taking the first character of the sequence and placing it at the end of the sequence (step 1).The characters ^ and $ mark the beginning and end of the sequence, respectively. Once these sequences are created, they are sorted (step 2). From this sorted matrix, the final column is selected as the transformed sequence (step 3). The transformed sequences is exactly the same length and has exactly the same characters as the original sequence, but in a different ordering. The sequence at the bottom is a longer sequence starting with the same 14-mer that demonstrates the effect on the transformed sequence of using a longer input sequence.

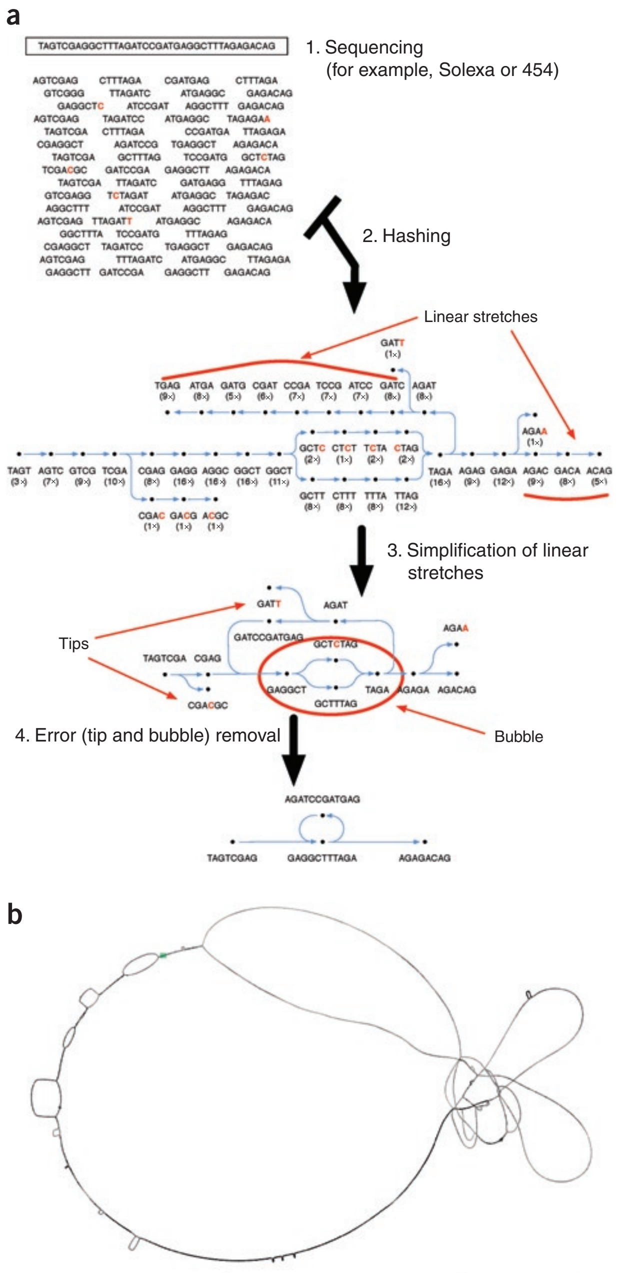
a
TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG
1. Sequencing
(for example, Solexa or 454)
2. Hashing
Linear stretches
3. Simplification of linear stretches
Tips
Bubble
4. Error (tip and bubble) removal
AGATCCGATGAG
TAGTCGAG
GAGGCTTTAGA
AGAGACAG
b

Figure 3 | Constructing and visualizing a de Bruijn graph of a DNA sequence. (**a**) An example de Bruijn graph assembly for a short genomic sequence without polymorphism. Sequence at top represents the genome, which is then sampled using shotgun sequencing in base space with 7-bp reads (step 1). Some of the reads have errors (red). In step 2, the *k*-mers in the reads (4-mers in this example) are collected into nodes and the coverage at each node is recorded. There are continuous linear stretches within the graph, and the sequencing errors create distinctive, low-coverage features through out the graph. In step 3, the graph is simplified to combine nodes that are associated with the continuous linear stretches into single, larger nodes of various *k*-mer sizes. In step 4, error correction removes the tips and bubbles that result from sequencing errors and creates a final graph structure that accurately and completely describes in the original genome sequence. (**b**) A full de Bruijn graph of two related plasmids that have a locus in common. The de Bruijn graph was created with 30-bp *k*-mers. The open loops are regions that differ between the two plasmids, whereas the heavier lines indicate common regions.

length from a reference assembly. The read lengths need only be over the *k*-mer length to generate a reasonable assembly (in theory, *k* must be over 15 bp, though in practice 19 is the lowest sensible *k*-mer, and larger *k*-mers are always better, although at the expense of having to generate more coverage to support these large *k*-mer sizes).

The first assembler to exploit this technology was Roche's 454 assembler, Newbler, which adapted the scheme specifically to handle the main source of error in 454 sequencing—namely, ambiguity in the length of homopolymer runs. In late 2007 and early 2008, several second-generation de Bruijn graph assemblers were released for very short reads, compatible with the Solexa technology, including SHARCGS[27], VCAKE[28], VELVET[29], EULER-SR[30], EDENA[31], ABySS[32] and ALLPATHS[33]. Some of these methods, such as VELVET, EULER-SR and ABySS, explicitly use de Bruijn graphs, whereas other methods implicitly explore a de Bruijn graph—for example, constrained by read-pair behavior, as in ALLPATHS. The methods differ in how they treat errors and to what extent they use read-pair information. Read pairs are defined as two short DNA sequence reads generated from different ends of a longer DNA molecule—for example, 35-bp reads generated from both ends of a 500 bp fragment. One does not know the identity of the sequence between the read pairs, but one usually has an estimate of the length of the intervening sequence. As it is only marginally more expensive to generate short reads in read-pair format than as single reads, extremely high coverage of read pairs is routinely available. The more advanced de Bruijn graph assemblers[29,30,32,33] can use read pairs to provide long assemblies. A particular challenge has been the two-base-encoding 'color space' of ABI SOLiD technology. In this two-base encoding, a single error produces a systematic translation error on all subsequent decoding of the bases for the rest of the read. In the context of an alignment, such an encoding scheme can be integrated into the alignment routine, and there is an argument that the double base encoding provides better discrimination between errors and observed differences. In *de novo* assembly, however, there is no reference. The solution has been to perform the assembly directly in color space and then 'key' the resulting color space assembly to one of the four feasible base-pair assemblies using either a small amount of traditional sequence or the presence of a known base at the start of each SOLiD read.

Whichever sequencing technology and assembly method are used, the ability to provide long assemblies critically requires that at least a proportion of the read pairs are longer than the longest common near-identical repeat in the genome. This varies considerably between genomes. Bacterial genomes often have only a handful of near-identical repeats longer than 200 bp (**Fig. 3**), whereas complex genomes, such as the human, usually have their repeat length determined by whether there has been an active LINE or SINE transposable element (usually around 4 kb in length for the former and between 500 bp and 1 kb for the latter). As the ability to produce longer read pairs (also referred to as 'mate pairs' to distinguish them from the shorter read pairs) has only recently been optimized for next-generation technologies, assemblies of complex genomes have been rare.

The other main barrier for large, complex genome assemblies is the memory overhead for these methods. Although the de Bruijn data structure is compressed, all the methods use some sort of adjunct data structures in addition to the core de Bruijn graph to map the reads to the graph. These adjunct structures are critical for leveraging additional information required for accurate assemblies, such as read pair information.

Many of the implementations that work well on a small scale (<50 -megabase genomes) would require over 2 terabytes of real memory for a complex genome. Several groups have tackled this engineering problem—in particular, the ABySS assembler, the SOAP assembler and the Cortex assembler (Z. Iqbal, M. Caccomo and P. Flicek, unpublished data). ABySS works using a message-passing interface (MPI)-cluster approach, whereas SOAP and Cortex use multiple passes over compressed data structures that can be retained on disk to handle the large data volumes. A key feature in all these methods is the early removal of sequencing errors; as each sequencing error produces usually a unique sequence, the full graph, with errors, is dominated by the behavior of errors, which produces a large number of extraneous of tips and bubbles in the graph. At present, these large scale methods are still in active development but they are likely to become more widespread and mature over 2010.

Discussion

Biologists interested in sequencing to answer their experimental questions should prepare themselves to join a fast-moving field and embrace the tools being developed specifically for it. As more sequence is generated, effective use of computational resources will be more and more important.

As a general recommendation, the goal for data analysis should be to effectively deal with as much data as quickly as possible. We are reminded of the importance of this with every announcement of greater sequencing capacity from the existing machines and with predicated capacities from sequencing technologies still under development. This means more flexible alignment pipelines will be the way of the future: the bulk of the reads will be aligned by a fast, less sensitive method and remaining reads will be aligned, possibly in several stages as dictated by the experimental design, by progressively slower and more sensitive methods on smaller and smaller subsets of the original collection of reads. This analysis technique has similarities to serial BLAST searching[34] but is conceptually revised to use multiple algorithms rather that multiple parameter sets for the same program.

As next-generation sequencing machines and their resultant piles of data have spread to many labs around the world, many groups have taken up the challenge of writing new algorithms specifically tuned for it. There are now dozens of published and unpublished short-read alignment programs and slightly fewer assembly programs, but for both types, more are appearing every month and only a few were used as examples to illustrate the methods in this review. However, the goal of this review is not to catalog the entire class of alignment and assembly tools, a task far better suited to the rapid update cycle of the World Wide Web. Indeed, dedicated pages on Wikipedia (http://en.wikipedia.org/wiki/List_of_sequence_alignment_software and http://en.wikipedia.org/wiki/Sequence_assembly) and on the community site SEQanswers (http://seqanswers.com/forums/showthread.php?t=43) at present provide the most current census and capabilities list of almost all of the existing alignment and assembly programs, including those mentioned here.

Looking ahead, a major goal will be to effectively apply sequencing to genomes without existing reference assemblies. Future developments of existing technologies, including increased read length, higher capacity and larger insert sizes, will immediately benefit the current generation of assembly algorithms. For forthcoming sequencing technologies, such as SMRT sequencing from Pacific Biosciences, increased read length is again a common theme. This may well see the reemergence of the read-based assemblers, but it is likely that the de Bruijn frameworks will remain a useful, if not optimal structure. Even with long read lengths, it is likely that high coverage techniques will be used, and a number of next-generation technologies have variable read length distributions, and can potentially trade off read length with sequence quality. The de Bruijn framework handles high coverage in a more compact form, and all the read-based methods can be executed in the context of a de Bruijn graph—the graph provides easy access to other features of the data; for example, for error correction. Whatever scheme is used, it is clear that yet more accurate assemblies can be achieved for more complex genomes with increased length in mate pairs for short-read technology or simply longer reads from new technologies.

Thus far, the development of short-read alignment programs has followed a predictable path. The first problem to solve was to ensure that the programs were sufficiently accurate and sufficiently fast. As more experience was gained and the properties of the sequence data became more clear, new algorithms emerged and existing ones were modified to incorporate new information about biases, sequence errors, regions of the genome inherently difficult to align and the effects of genome polymorphism both at the level of single base changes and of small indel and larger structural variations. At the same time, a community effort created common and optimized file formats for storage and exchange of the resulting alignment data[35]. As all this has been happening, the sales of sequencing machines have continued to grow quickly and the average amount of data produced by each machine has grown several times. These developments have led to the current push to ensure that the programs for creating accurate alignments that address all of the above issues are simply as fast as possible. New data types such as the interrupted read sequences from Complete Genomics will challenge the existing alignment algorithms, as will the increase in read length and experimental studies that focus on cancer genomes with multiple deletions, duplications and rearrangements. These problems are probably not yet solved, but the techniques described above will provide the first approach to these problems and the foundations for new approaches.

ACKNOWLEDGMENTS

The authors acknowledge D. Zerbino and support by the Wellcome Trust and the European Molecular Biology Laboratory.

COMPETING INTERESTS STATEMENT

The authors declare no competing financial interests.

Published online at http://www.nature.com/naturemethods/.
Reprints and permissions information is available online at http://npg.nature.com/reprintsandpermissions/.

1. Medvedev, P., Stanciu, M. & Brudno, M. Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods* **6**, S13–S20 (2009).
2. Pepke, S., Wold, B. & Mortazavi, A. Computational approaches to the analysis of ChIP-seq and RNA-seq data. *Nat. Methods* **6**, S22–S32 (2009).
3. Boyle, A.P. *et al.* High-resolution mapping and characterization of open chromatin across the genome. *Cell* **132**, 311–322 (2008).
4. Margulies, M. *et al.* Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437**, 376–380 (2005).
5. McKernan, K.J. *et al.* Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding. *Genome Res.* **19**, 1527–1541 (2009).
6. Bentley, D.R. *et al.* Accurate whole human genome sequencing using reversible terminator chemistry. *Nature* **456**, 53–59 (2008).
7. Batzoglou, S. The many faces of sequence alignment. *Brief Bioinform.* **6**, 6–22 (2005).

8. Li, H., Ruan, J. & Durbin, R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.* **18**, 1851–1858 (2008).
9. Li, R., Li, Y., Kristiansen, K. & Wang, J. SOAP: short oligonucleotide alignment program. *Bioinformatics* **24**, 713–714 (2008).
10. Rumble, S.M. *et al.* SHRiMP: accurate mapping of short color-space reads. *PLOS Comput. Biol.* **5**, e1000386 (2009).
11. Lin, H., Zhang, Z., Zhang, M.Q., Ma, B. & Li, M. ZOOM! Zillions of oligos mapped. *Bioinformatics* **24**, 2431–2437 (2008).
12. Ma, B., Tromp, J. & Li, M. PatternHunter: faster and more sensitive homology search. *Bioinformatics* **18**, 440–445 (2002).
PatternHunter was the first alignment program to implement the method of finding alignments by scanning with 'spaced seeds' that require exact matching positions to seed the alignments but do not require these seeds to be consecutive. This method is extremely effective for the mapping short sequencing reads and has been adopted by most hash-based alignment methods.
13. Rasmussen, K.R., Stoye, J. & Myers, E.W. Efficient q-gram filters for finding all epsilon-matches over a given length. *J. Comput. Biol.* **13**, 296–308 (2006).
14. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S.L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.* **10**, R25 (2009).
15. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009).
16. Li, R. *et al.* SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics* **25**, 1966–1967 (2009).
17. Burrows, M. & Wheeler, D.J. A block-sorting lossless data compression algorithm. Technical report 124, Digital Equipment Corporation (1994).
18. Ferragina, P. & Manzini, G. Opportunistic data structures with applications; doi:10.1109/SFCS.2000.892127 in *Proceedings of the 41st Symposium on Foundation of Computer Science (FOCS 2000)* 390–398 (IEEE Computer Society, 2000).
The FMindex of the BWT sequence first described in this paper is the fundamental result that has been leveraged by each of BWT-based alignment programs. The sequencing matching algorithm described here has been incorporated into each of the methods, with extensions to handle the specific problems of mismatches, gaps and paired reads.
19. Gräf, S. *et al.* Optimized design and assessment of whole genome tiling arrays. *Bioinformatics* **23**, i195–i204 (2007).
20. Kärkkäinen, J. Fast BWT in small space by blockwise suffix sorting. *Theor. Comput. Sci.* **387**, 249–257 (2007).
21. Flicek, P. The need for speed. *Genome Biol.* **10**, 212 (2009).
22. Staden, R. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Res.* **6**, 2601–2610 (1979).
23. Staden, R., Beal, K.F. & Bonfield, J.K. in Computer methods in molecular biology. in *Bioinformatics Methods and Protocols* vol. 132 (eds. Misener, S. & Krawetz, S.A.) 115–130 (Humana, Totowa, New Jersey, USA, 1998).
24. Pevzner, P.A., Borodovsky, M.Y. & Mironov, A.A. Linguistics of nucleotide sequences. II: Stationary words in genetic texts and the zonal structure of DNA. *J. Biomol. Struct. Dyn.* **6**, 1027–1038 (1989).
25. Idury, R.M. & Waterman, M.S. A new algorithm for DNA sequence assembly. *J. Comput. Biol.* **2**, 291–306 (1995).
Idury and Waterman first presented the fundamental algorithm for sequence assembly by k-mer extension. The representation of algorithm with the de Bruijn graph data structure is at the heart of the assembly method described here.
26. Pevzner, P.A. & Tang, H. Fragment assembly with double-barreled data. *Bioinformatics* **17** (suppl. 1), S225–S233 (2001).
27. Dohm, J.C., Lottaz, C., Borodina, T. & Himmelbauer, H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Res.* **17**, 1697–1706 (2007).
28. Jeck, W.R. *et al.* Extending assembly of short DNA sequences to handle error. *Bioinformatics* **23**, 2942–2944 (2007).
29. Zerbino, D.R. & Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.* **18**, 821–829 (2008).
30. Chaisson, M.J. & Pevzner, P.A. Short read fragment assembly of bacterial genomes. *Genome Res.* **18**, 324–330 (2008).
31. Hernandez, D., François, P., Farinelli, L., Osterås, M. & Schrenzel, J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res.* **18**, 802–809 (2008).
32. Simpson, J.T. *et al.* ABySS: a parallel assembler for short read sequence data. *Genome Res.* **19**, 1117–1123 (2009).
33. Butler, J. *et al.* ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res.* **18**, 810–820 (2008).
34. Korf, I. Serial BLAST searching. *Bioinformatics* **19**, 1492–1496 (2003).
35. Li, H. *et al.* The Sequence Alignment/Map (SAM) format and SAMtools. *Bioinformatics* **25**, 2078–2079 (2009).

Corrigendum: Sense from sequence reads: methods for alignment and assembly

Paul Flicek & Ewan Birney

***Nat. Methods* 6, S6–S12 (2009); published online 15 October 2009; corrected after print 6 May 2010.**

In the version of this article initially published online, the caption to Figure 3b was mislabeled. It shows a de Bruijn graph of two plasmids partially overlapping in sequence. The error has been corrected in the HTML and PDF versions of the article.