Bioinformatics, 34(10), 2018, 1659–1665
doi: 10.1093/bioinformatics/btx753
Advance Access Publication Date: 24 November 2017
Original Paper

Sequence analysis

Mapping-free variant calling using haplotype reconstruction from k-mer frequencies

Peter A. Audano*, Shashidhar Ravishankar and Fredrik O. Vannberg*

School of Biology, Georgia Institute of Technology, Atlanta, GA 30332, USA

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on June 21, 2017; revised on November 11, 2017; editorial decision on November 15, 2017; accepted on November 23, 2017

Abstract

Motivation: The standard protocol for detecting variation in DNA is to map millions of short sequence reads to a known reference and find loci that differ. While this approach works well, it cannot be applied where the sample contains dense variants or is too distant from known references. *De novo* assembly or hybrid methods can recover genomic variation, but the cost of computation is often much higher. We developed a novel k-mer algorithm and software implementation, Kestrel, capable of characterizing densely packed SNPs and large indels without mapping, assembly or de Bruijn graphs.

Results: When applied to mosaic penicillin binding protein (PBP) genes in *Streptococcus pneumoniae*, we found near perfect concordance with assembled contigs at a fraction of the CPU time. Multilocus sequence typing (MLST) with this approach was able to bypass *de novo* assemblies. Kestrel has a very low false-positive rate when applied to the whole genome, and while Kestrel identified many variants missed by other methods, limitations of a purely k-mer based approach affect overall sensitivity.

Availability and implementation: Source code and documentation for a Java implementation of Kestrel can be found at https://github.com/paudano/kestrel. All test code for this publication is located at https://github.com/paudano/kescases.

Contact: paudano@gatech.edu or fredrik.vannberg@biology.gatech.edu

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

1 Introduction

Although modern alignment tools are designed to handle errors and variation, a sequence read that differs significantly from the reference cannot be confidently assigned to the correct location. When these reads are mapped, the low alignment confidence leads to low variant call confidence, and it becomes difficult to separate true variants from false calls. In some regions, the read may be clipped or not mapped at all. As a result, variant calling algorithms that rely solely on alignments cannot characterize these events.

Bacterial genomes are small and relatively simple, but they remain one of the hardest informatic targets due to their variability. Horizontal gene transfer (HGT) is one mechanism that changes samples significantly when compared to the reference, and it often leads to drug resistance or increased virulence (Ochman *et al.*, 2000). Bacterial typing and surveillance methods often require whole genome assembly (Thomsen *et al.*, 2016), a database of known alleles (Li *et al.*, 2016) or multiple reference sequences (Li *et al.*, 2012). These approaches add to the time required to run the analysis and to maintain allele databases. The ideal method would handle bacterial diversity with a single reference, and it would be capable of placing genomic alterations in a uniform context without switching references.

There are several alternative methods commonly employed, but they are either limited in power or consume far more computing resources than the alignment and variant-calling protocol. Calling variants from *de novo* assembled contigs may work for monoploid organisms, but since it reduces the read coverage to 1, false-calls cannot easily be corrected (Olson *et al.*, 2015). Building the de Bruijn graphs that are typically employed in assembly may also require many gigabytes of memory or computing clusters (Li *et al.*, 2010). Cortex (Iqbal *et al.*, 2012) was designed to overcome limitations of the *de novo* assembly approach, but it still relies on de Brujin graphs assembled over the whole genome. Platypus (Rimmer *et al.*, 2014) performs local assemblies, however, it incurs the overhead of both read mapping and local assembly if variant loci are not known *a priori*.

One possible approach is to use the information contained within a set of k-mer frequencies over the sequence data. K-mers can be represented numerically, they do not rely on sequence alignments, and k-mer counting methods are fast. To date, sparsely spaced SNPs have been corrected with such an approach (Gardner and Slezak, 2010; Gardner *et al.*, 2013), but a robust variant calling algorithm has never been shown to work. Because dense SNPs and large insertions can create many k-mers very different from any reference sequence, how such an approach could work efficiently is not immediately clear.

We constructed an algorithm that relies strictly on k-mer frequencies over sample sequence data and ordered k-mers in a reference sequence to call variants. This method first finds regions of variation over the reference using disruptions in the frequency distribution of expected k-mers, and by beginning with unaltered k-mers at the flanks of such regions, it can greatly simplify the search through k-mer space while resolving variation. Using this approach, we show how Kestrel can re-build altered regions of the genome in a targeted manner and resolve regions of dense variation.

2 Materials and methods

2.1 Algorithm overview

Using features in KAnalyze (Audano and Vannberg, 2014) version 2.0.0, the frequency of each k-mer in the sample is stored in an indexed k-mer count file (IKC) (Fig. 1a). The IKC records are grouped by a k-mer minimizer (Roberts *et al.*, 2004) and sorted. This structure tends to cluster k-mers from the original sequence, and by reading it as a memory mapped file, k-mers can be rapidly queried with low memory requirements using search optimizations similar to those implemented in Kraken (Wood and Salzberg, 2014).

From the reference sequence, the frequencies for each k-mer and its reverse complement are obtained from the IKC file and summed, but these are left in order (Fig. 1b). Kestrel searches the resulting array for loci where the frequency declines and recovers (Fig. 1c), which suggests the k-mers of the reference and sample differ. Analogous to the GATK (McKenna *et al.*, 2010) HaplotypeCaller, this low-frequency region is called an *active region*, and *haplotypes* are reconstructed over it.

Starting with the high-frequency k-mer on the left end of the active region, the first base is removed, all four bases are appended to this (k - 1)-mer, and the k-mer frequency is queried for each of the four possibilities (Fig. 1d). An equivalent process is performed on the reverse complement k-mer, and the frequencies are summed. The base that yields a high frequency k-mer is appended to the haplotype. The new k-mer ending in that base is then used to find the next base by the same process. If more than one of these k-mers has a high frequency, a haplotype is assembled for each one.

A modified Smith-Waterman (Smith and Waterman, 1981) algorithm guides the process by aligning the active region and the haplotype as it is reconstructed (Fig. 1e). By setting an initial score and disallowing links to zero score states, alignments are anchored on the left and are allowed to extend until an optimal alignment is obtained. Variant calls follow trivially from the alignment (Fig. 1f and g).

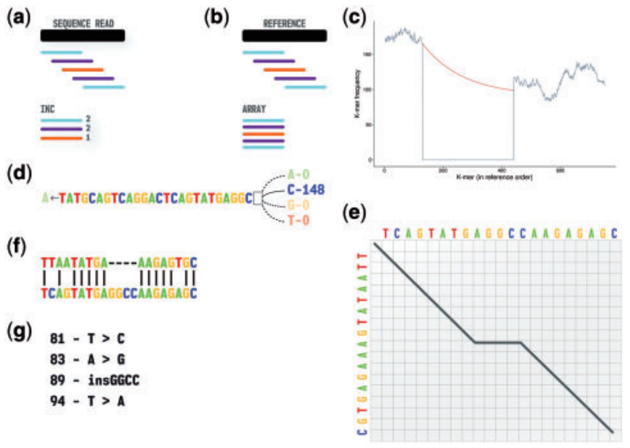

Fig. 1. Overview of the Kestrel process from sequence data to variant call. (**a**) The sequence reads are converted to an IKC file. (**b**) The reference sequence is converted into an array of k-mers and left in reference order. (**c**) K-mer frequencies from the sequence reads (vertical axis) are assigned to the ordered k-mers of the reference (horizontal axis). A decline and recovery of the frequencies bound an active region where one or more variants are present. The recovery threshold is degraded with an exponential decay function (red) to allow for declining read coverage. (**d**) Starting from the left anchor k-mer (last k-mer with a high frequency), the first base is removed, each possible base is appended, and the base that recovers the k-mer frequency is appended to the haplotype. (**e**) A modified alignment algorithm tracks haplotype reconstruction and terminates the process when an optimal alignment is reached. (**f**) This algorithm yields an alignment of the reference sequence and haplotype within the active region. (**g**) Variant calls are extracted from the alignment (Color version of this figure is available at *Bioinformatics* online.)

By relaxing these criteria and by building in either the forward or reverse direction, it is possible to call variants up to either end of the reference.

2.2 Active region detection

The distribution of k-mer frequencies from a set of sequence data is approximately uniform with a mean equal to the average read depth of the sample, and variants disrupt this distribution over reference k-mers. For example, a single SNP causes k k-mers of the sample to differ from the reference. Active region detection searches for differences between neighbors that are less likely to have occurred by chance. Then by searching for a recovery in the downstream frequencies, it attempts to resolve the left and right breakpoints in k-mer space. This region is bounded by the unaltered k-mers at its flanks, which we label *anchor k-mers*. The low-frequency and the anchor k-mers together comprise the active region.

The frequency difference between of two neighboring k-mers, N_i and N_{i+1}, is evaluated and compared to ϵ, the threshold required to trigger an active region scan ($|N_i - N_{i+1}| > \epsilon$). Because read depth can vary greatly over samples, setting this parameter to some value is unlikely to perform well on real sequence data. Therefore, Kestrel sets ϵ to some quantile, Q_ϵ, of the set of $|N_i - N_{i+1}|$ for all neighboring k-mers in the reference. The default quantile, 0.90, with an absolute minimum of 5 was found to work well in practice. Supplementary Section S1.3 discusses active region detection in more detail.

Sequencing errors, PCR duplicates, GC biases and other factors systematically work to disrupt the uniformity of the frequency distribution. Therefore, any robust approach must be equipt with heuristics to work around such biases. Kestrel uses an exponential decay function over the recovery threshold, and it ignores short peaks in the frequency distribution.

When an active region occurs over declining read coverage, the recovery k-mer frequency may be lower than expected. As the scan moves further from the left anchor k-mer, the expected recovery frequency should be relaxed. The exponential decay function, $f(x)$ (Equation 1), is employed to reduce the recovery threshold as the active region extends. $f(0)$ is the anchor k-mer frequency, and it approaches a lower bound, f_{min}, asymptotically. By default, $f_{min} = 0.55 \cdot f(0)$ to avoid ending an active region prematurely on a large heterozygous variant region. $f(x)$ is defined by scaling and shifting the standard exponential decay function, $h(x)$ (Equation 2).

$$f(x) = (f(0) - f_{min}) \cdot h(x) + f_{min} \tag{1}$$

$$h(x) = e^{-x\lambda} \tag{2}$$

$h(x)$ is parameterized by λ, which must be also be set, but Kestrel does not configure this parameter directly because it is difficult to know how to choose a reasonable value. Instead, λ is chosen by a configurable parameter, α, that is defined as the proportion of the decay range, $f(0) - f_{min}$, after 1 k-mer (Equation 3). This provides a more intuitive way to define how rapidly the recovery threshold is allowed to decline.

$$\begin{aligned} h(k) &= \alpha \\ e^{-k\lambda} &= \alpha \\ -k\lambda &= \log(\alpha) \\ \lambda &= \frac{-\log(\alpha)}{k} \end{aligned} \tag{3}$$

At k k-mers, the recovery threshold $f(k) = \alpha \cdot (f(0) - f_{min}) + f_{min}$. In other words, $f(k)$ has declined in its range from $f(0)$ to f_{min} by a factor of α. This is true for all nk such that $f(nk) = \alpha^n \cdot (f(0) - f_{min}) + f_{min}$ (Equation 4). Supplementary Section S2.2 outlines active region heuristics in more detail.

$$\begin{aligned} h(x) &= e^{-x\lambda} \\ &= e^{-x\frac{-\log(\alpha)}{k}} \\ &= \left(e^{\log(\alpha)}\right)^{\frac{x}{k}} \\ &= \alpha^{\frac{x}{k}} \qquad \square \end{aligned} \tag{4}$$

2.3 Haplotype alignment

After the endpoints of an active region are found, the actual sequence (haplotype) from the sample must be reconstructed. This process is similar to a local assembly from k-mers, but it is implemented to keep resource usage at a minimum. Since the reference sequence over the whole region is known and anchor k-mers have been chosen, the search through k-mers in the sample can be greatly simplified.

The process begins by initializing the haplotype with the left anchor k-mer. Then by removing the left-most base of the k-mer and appending a new base to the end, the k-mer can be shifted one base to the right. Each nucleotide can be appended and the frequency checked. The base that produces a k-mer with the highest frequency is appended to the haplotype. If more than one base produces an acceptable frequency, then the alignment is split by saving the state of reconstruction with alternate bases, continuing with the highest-frequency k-mer, and returning to the alternatives after the current haplotype is built. In this way, multiple haplotypes may be built over one active region.

While this algorithm is simple enough to build the haplotype sequence, it does not know when to terminate. It could continue until the right anchor k-mer is found, but such reference-naïve reconstruction would certainly waste many CPU cycles extending erroneous sequence. Because the active region sequence and the haplotype sequences are known, an alignment could be performed as it is extended. A global alignment would be ideal except that it would have to be recomputed each time a base is added, and performance of such an algorithm would be unacceptable in all but the most trivial of applications.

An ideal algorithm would align over the active region from end to end, anchor the left ends of the haplotype and active region, and allow the right end of the haplotype to extend. To accomplish this, Kestrel employs a modified Smith-Waterman alignment. Two key modifications were made to Smith-Waterman; (i) any subalignment with a score of 0 cannot be extended, and (ii) the alignment must begin with a score greater than 0. Because of these modifications, the alignment must begin with a non-zero score, and the gap extension penalty must be non-zero to prevent unbounded extension.

Smith-Waterman is a dynamic programming approach where matrices track scores by updating from shorter sub-alignments as the alignment progresses (Eddy, 2004). The active region is positioned over the vertical axis, and the haplotype is positioned over the horizontal axis of the score matrix. We define the active region base in row i as x_i and the haplotype base in column j as y_j. As haplotype bases are added to the alignment, a column is appended to the matrix. Section 2.6 discusses how this is done efficiently.

For aligned bases, R_{match} is added to the score if they agree and $R_{mismatch}$ is added if they disagree. For convenience, we define $match(i, j)$ (Equation 5) to return R_{match} if x_i and y_j match, or $R_{mismatch}$ if they do not. This implementation employs an affine gap model that allows for distinct gap open (R_{open}) and gap extension (R_{gap}) penalties. This type of model requires three matrices over x and y to track the scores. One matrix, S_{aln}, contains scores through aligned (matched or mismatched) bases. Two more score matrices, S_{gact} and S_{ghap}, contain scores through gaps in the active region and gaps in the haplotype, respectively. Using these definitions, the modified Smith-Waterman score update process is defined in Equations 6–8.

$$match(i,j) = \begin{cases} R_{match} & : x_i = y_j \\ R_{mismatch} & : x_i \neq y_j \end{cases} \tag{5}$$

$$S_{aln}(i,j) = \max \begin{cases} 0 \\ S_{aln}(i-1,j-1) + match(i,j) & : \\ \qquad S_{aln}(i-1,j-1) > 0 \\ S_{gact}(i-1,j-1) + match(i,j) & : \\ \qquad S_{gact}(i-1,j-1) > 0 \\ S_{ghap}(i-1,j-1) + match(i,j) & : \\ \qquad S_{ghap}(i-1,j-1) > 0 \end{cases} \tag{6}$$

$$S_{gact}(i,j) = \max \begin{cases} 0 \\ S_{aln}(i,j-1) + R_{open} + R_{gap} & : \\ \qquad S_{aln}(i,j-1) > 0 \\ S_{gact}(i,j-1) + R_{gap} & : \\ \qquad S_{gact}(i,j-1) > 0 \\ S_{ghap}(i,j-1) + R_{open} + R_{gap} & : \\ \qquad S_{ghap}(i,j-1) > 0 \end{cases} \tag{7}$$

$$S_{ghap}(i,j) = \max\begin{cases} 0 \\ S_{aln}(i-1,j) + R_{open} + R_{gap} & : \\ \quad S_{aln}(i-1,j) > 0 \\ S_{gact}(i-1,j) + R_{open} + R_{gap} & : \\ \quad S_{gact}(i-1,j) > 0 \\ S_{ghap}(i-1,j) + R_{gap} & : \\ \quad S_{ghap}(i-1,j) > 0 \end{cases} \quad (8)$$

To initialize the alignment, the bases of the active region are positioned along the vertical axis of the matrices, and each base of the anchor k-mer creates one column. The first base of the sequences is in row and column 1. Row and column 0 exist for convenience, but they do not align any bases. The initial alignment score, R_{init} is assigned ($S_{aln}(i,i) = R_{init}\,, 0 \le i \le k$) where k is the size of k-mers. All other scores in all three matrices are initialized to 0. A fourth matrix, T, contains traceback information from the end of an alignment to $S_{aln}(0,0)$. It is initialized so that $T(i,i) \rightarrow T(i-1,i-1)\,, 0 < i \le k$. This initialization of S_{aln} and T creates a single path for the anchor k-mer in the alignment, and all acceptable alignments must enter this path at $S_{aln}(k,k)$. The alignment extends from the anchor k-mer as already described. Supplementary Section S1.4-7 outlines the alignment process and data structures in more detail.

2.4 Alignment termination

Since the alignment must cover all of the active region, only the last row of S_{aln} needs to be queried to find the best score for the current alignment, R_{max} (Equation 9). The maximum potential score that might be obtained by adding more bases is determined by examining the last column of S_{aln}. The best possible score from $S_{aln}(i,j)$ is the case where all subsequent bases of the active region are aligned with matched based, *maxpot*(i, j) (Equation 10), and R_{maxpot} is the maximum of *maxpot*(i, j) (Equation 11). If $R_{max} > R_{maxpot}$, haplotype extension terminates. Supplementary Section S1.8-9 further outlines optimal scores and alignment termination.

$$R_{max} = \max\{S_{aln}(|x|,j), 0 \le j \le |y|\} \quad (9)$$

$$maxpot(i,j) = \begin{cases} 0 : \\ \quad S_{aln}(i,j) = 0 \\ S_{aln}(i,j) + (|x| - i) \times R_{match} : \\ \quad S_{aln}(i,j) > 0 \end{cases} \quad (10)$$

$$R_{maxpot} = \max(\{maxpot(i,|y|), 0 \le i \le |x|\}) \quad (11)$$

2.5 Variant calling

The variants are interpreted from the alignment, and the locations of the variants are translated with respect to the location of the active region. The alignment provides a convenient way to identify mismatched bases and gaps in either sequence.

If any cell of trace matrix, T, has more than one path out, then there is more than one optimal alignment, and so there are multiple ways to translate the alignment to variant calls. When comparing two alignments, the one with the first non-matching base is given a higher priority. If the non-matching bases agree (same variant), then the next non-matching base is queried. If the non-matching bases do not agree, then alignments are prioritized by mismatch, insertion and deletion, in that order. This gives the algorithm predictable output for cases such as a deletion in a homopolymer repeat; Kestrel will always report that the first base was deleted even though the alignment score would be the same for a deletion at any locus of the repeat. The effect is to left-align variant calls.

Approximate read depth of an active region is estimated by summing the depth of all haplotypes, which may include the reference haplotype. Similarly, summing only the haplotypes that support a variant call gives the approximate depth of the variant. Because haplotypes may share k-mers, we use the minimum k-mer frequency as a conservative estimate of read depth of any single haplotype. These estimates may be useful for filtering variants with low support.

2.6 Alignment implementation

Because of the nature of the dynamic programming algorithm, only the last column of the score matrices (S_{aln}, S_{gact} and S_{ghap}) needs to be stored while the next column is built. Therefore, each of these matrices can be reduced to two arrays where one contains the last column, and one contains the new column being added. When alternate haplotypes are explored and the alignment splits, only one array for each matrix must be saved.

The traceback matrix, T, is more complex because it is not stored as a matrix. Instead, it is a linked-list of alignment states that always leads back to $S_{aln}(0,0)$. When a non-zero score is added to a score matrix, a link is added to T. For each non-zero score in score matrices, a link from the matrix to a node in T is stored.

Since T is a linked list that is only traversed toward $S_{aln}(0,0)$, one node of the alignment may have several links into it. Therefore, different haplotypes may link to the same node in T where they split and no part of T needs to be duplicated. Both haplotypes will trace back to the point where they diverged and continue toward S_{aln} $(0,0)$ along the same path.

The linked list structure has another more subtle property that Java uses to keep memory usage low. When an alignment path reaches a dead end, the node at the end of the path has no reference to it. This allows Java Virtual Machine (JVM) garbage collection (GC) to detect and remove these nodes. In other words, GC can automatically prune dead branches of T. This improves scalabilty by reducing the memory requirements for large active regions where many haplotypes are investigated.

3 Results

3.1 *S.pneumoniae* test case

We analyzed the four penicillin binding protein (PBP) genes of *Streptococcus pneumoniae* (*S.pneumoniae*) targeted by β-lactam compounds. According to several studies, 20% or more of a PBP gene may be altered by inter-species recombination (Laible *et al.*, 1991; Martin *et al.*, 1992), and this can create mosaic PBP genes with a lower β-lactam binding affinity. Because these recombination events often alter hundreds of contiguous bases, variant calling from a standard alignment pipeline cannot characterize them.

We obtained 181 samples from 29 serotypes recently released by the Centers for Disease Control and Prevention from NCBI under BioProject PRJNA284954 (SRR2072210-SRR2072387, SRR2076738-SRR2076740). These data are whole-genome 250 bp paired-end Illumina sequence reads ranging from 15 Mbp to 1234 Mbp (median = 323 Mbp). Selecting the best reference out of more than 10 is often necessary (Li *et al.*, 2012), however, we tested Kestrel's ability to characterize variants using a single reference for all samples, TIGR4 (NC_003028.3).

We called all variants in four PBP genes (PBP2X, PBP1A, PBP2B and PBP2A) using three distinct approaches. The first is a standard alignment pipeline using BWA (Li and Durbin, 2009, 2010), Picard and GATK (McKenna *et al.*, 2010) HaplotypeCaller. The second is Kestrel. The third is a *de novo* assembly pipeline using SPAdes (Bankevich *et al.*, 2012), BWA and SAMtools (Li *et al.*, 2009). Variants identified by the assembly where the depth of aligned contigs is 1 were defined as the true variants. HaplotypeCaller and Kestrel variant calls were compared to the true set with RTG (Cleary *et al.*, 2015) vcfeval.

Contamination in the PBP genes was identified in SRR2072298, SRR2072306, SRR2072339, SRR2072342, SRR2072351 and SRR2072379, and these samples are not included in our analysis results. For all of these samples, there were many variant calls in the PBP genes with a relative depth of 0.70 or less, which is unlikely in a monoploid organism. The size of the IKC files is also larger than expected, which supports our conclusion that they contain allelic variation not present in the host genome (Fig. 2). Based on the IKC files, two more samples (SRR2072345 and SRR2072352) may also contain non-host or extra-chromosomal material, but since we saw no evidence for this in the variant calls over the PBP genes, these samples were included. Two other samples (SRR2072219 and SRR2072360) did not have complete sequence data and were also removed. They contain 10 Mbp and 37 Mbp, respectively, where the median has 323 Mbp and the next lowest sample has 92 Mbp. Supplementary SectionS 3.4 discusses how removed samples were identified.

With the minimum k-mer frequency set to 5, sequence read errors are filtered out of the distribution of k-mers. It is then interesting to note that a sample has a finite set of solid k-mers, and once this set is reached, the IKC file ceases to grow in size. BAM files must store a record for each read, and so they grow approximately linearly with read depth. The samples suspected of contamination and low coverage indeed show an increase and a decrease, respectively, in IKC file size (Fig. 2).

Kestrel produced 29 806 true positive (TP) variant calls with 73 false positive (FP) and 100 false negative (FN) calls (Fig. 3a) (sensitivity = 1.00, FDR = 0.00). Because mosaic regions disrupted the alignments, GATK was only able to produce 17 636 TP variant calls with 12 FP and 11 777 FN calls (Fig. 3b) (sensitivity = 0.60, FDR = 0.00). GATK tends to represent dense SNPs as insertion/deletion pairs, and so the expected true variants differ between the two approaches. This is a diverse set of samples varying in their relationship with the reference and mosaic content, and a few samples contribute most of the variants (Fig. 3c).

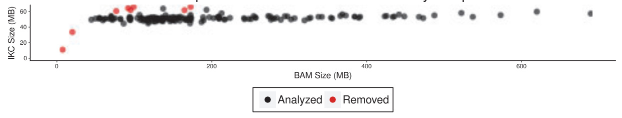

Fig. 2. Size of BAM files vs IKC files for each sample. Since low frequency k-mers are removed, the size of an IKC file does not continue to grow with read depth once a full set of representative k-mers are present. Since BAM files have a record for each read, their size does increase with the number of reads. Samples removed for suspected contamination and low coverage are shown in red. Low-coverage samples lack a representative set of k-mers at sufficient frequency, and so the file size falls below the distribution. Samples with contamination contain k-mers that do not belong to the sample, and so their size rises above the distribution (Color version of this figure is available at *Bioinformatics* online.)

Kestrel required an average of 13.3 CPU minutes per million 250 bp reads (min/M-reads), the alignment approach required an average of 14.5 min/M-reads, and the assembly approach required 106.2 min/M-reads (Fig. 3d). For all steps of the pipeline, Kestrel consumed an average of 1.0 GB of memory, GATK consumed an average of 2.7 GB, and the assembly approach consumed 9.1 GB. Maximum memory consumption was 2.3, 3.6 and 15.7 GB respectively for Kestrel, GATK and assembly (Fig. 3e). Runtime metrics were obtained on a 12 core machine (2 x Intel Xeon E5-2620) with 32 GB of RAM (DDR3-1600), RAID-6 over SATA drives (3 GB/s, 72 K RPM) and CentOS 6.7.

3.2 MLST test case

Multilocus Sequence Typing (MLST) attempts to cluster samples by analyzing the content of some set of genes. The current method is to build a BLAST (Altschul *et al.*, 1990) database from *de novo* assembled contigs and to use the database for comparing each allele to the sequence data (Jolley and Maiden, 2010; Larsen *et al.*, 2012). A whole-genome assembly is an expensive operation, so we tested Kestrel's ability to bypass it.

We obtained 7 *Neisseria meningitidis* (*N.meningitidis*) samples from ENA study PRJEB3353 (ERR193671-ERR193677) (Reuter *et al.*, 2013) and allele sequences for 7 house-keeping genes (adk, aroE, abcZ, fumC, gdh, pgm and pdhC) from pubMLST (Jolley and Maiden, 2010). These data are whole-genome 150 bp paired-end Illumina reads. For each sample, Kestrel identified the best allele by using each allele as a reference sequence.

Sequence reads were assembled with SPAdes using default options, and contigs were used to construct a BLAST database. Each allele downloaded from pubMLST for *N.meningitidis* was used as a BLAST query, and the best allele for each gene was chosen. With the allele calls, the sequence type (ST) was identified by comparing against sequence type profiles also obtained from pubMLST. This set of alleles and the sequence type represents the expected results based on the current methods.

Sequence reads were also transformed to an IKC file with KAnalyze using 31-mers, a 15-base minimizer, a minimum k-mer frequency of 5 and a minimum allele depth of 0.50. Each allele was used as a reference sequence for variant calling. For each gene, the allele with the fewest variants was chosen as the best match. If no variants were detected for an allele, the k-mer counts over the allele reference was checked to ensure that it was present in the sequence data. The best allele matches were used to identify the ST by comparing them against types from pubMLST.

There was 100% concordance between the assembly and Kestrel methods. All samples were called with a minimum k-mer frequency of 5 except ERR193672, which was reduced to 2 to call gene pgm because of low coverage.

The assembly-based MLST approach required an average of 51.6 min/M-reads and average of 7.1 GB of memory (max 7.8 GB). The Kestrel MLST approach required 10.4 min/M-reads and 1.6 GB of memory (max 1.7 GB).

3.3 *E.coli* test case

We tested Kestrel's ability to call variants on whole genomes using 309 *Escherichia coli* (*E.coli*) samples obtained from assembled contigs in the supplementary information published by Salipante *et al.* (2015) (Dataset S7.tar.gz). The study reports on 312 samples, but two were not found in the online dataset (upec-240 and upec-52) and one consistently caused RTG vcfeval to crash (upec-9).

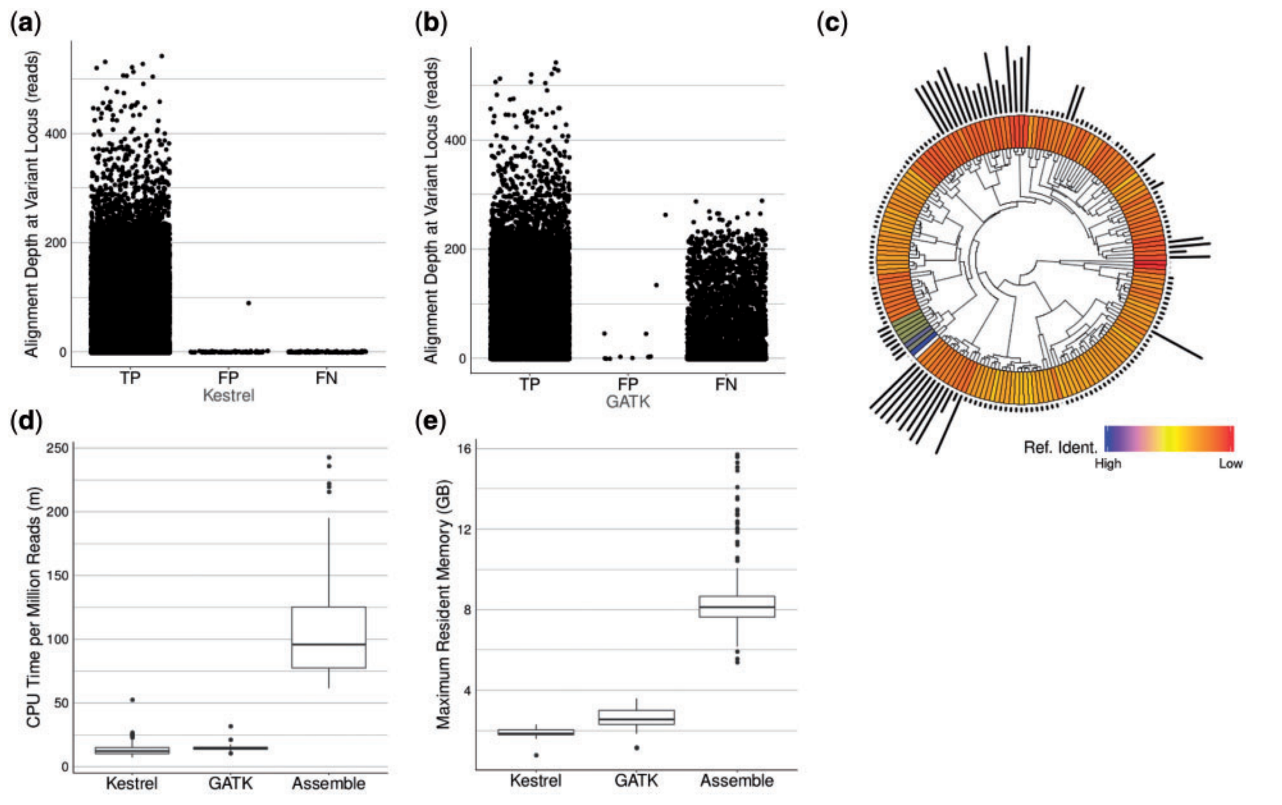

Fig. 3. Results of testing Kestrel and the alignment approach using GATK over 173 *S.pneumoniae* samples. (**a**) Variant call summary for Kestrel calls depicting TP, FP, FN calls. (**b**) Variant call summary for GATK calls depicting TP, FP and FN calls. (**c**) A plot depicting the phylogeny of all samples by ANI (inner track), the distance from the reference (white) by ANI as a blue-yellow-red heatmap (middle track), and the relative number of variants in each sample (outer track). (**d**) CPU minutes per million reads. When an assembly is required, the required CPU time increases. (**e**) Maximum memory usage in gigabytes (GB) for each pipeline (Color version of this figure is available at *Bioinformatics* online.)

The contigs were aligned to an *E.coli* K-12 reference (NC_000913.3), and variants between the reference and contig were identified. To minimize the effect of assembly errors, we used ART (Huang *et al.*, 2012) with contig sequences to simulate 150 bp paired-end reads to an average depth of 30×. Kestrel and GATK were run on the 4.6 Mbp reference using the same tools and approach as the *S.pneumoniae* contigs where the contig depth was at least 1 (mean 4.0 Mbp).

KAnalyze and Kestrel were applied to these simulated reads with the same parameters as the *S.pneumoniae* reads (k-mer size 31, quality 30, min depth 5). An additional parameter was given to Kestrel to discard variants over ambiguous reference bases, such as N, so that RTG vcfeval had an equivalent set of variants from both approaches. With all variant call sets, RTG vcfeval was used to test the calls. Kestrel must limit the size of active regions when calling on whole genomes or it will try to resolve erroneously large regions and perform poorly. For larger genomes, increasing the k-mer size may help reduce k-mer distribution noise.

In addition to a VCF, Kestrel can output a SAM file of the haplotypes it assembles and aligns. The SAM feature was enabled for this experiment so that the locations of haplotypes could be identified. Variant statistics were calculated twice; once over all regions where the alignment depth was one and once for all regions where there was an assembled haplotype. The current implementation of Kestrel discards active regions with one wildtype haplotype, so the second analysis only applies to regions where at least one variant was identified (mean 2.3 Mbp).

Over the whole genome, Kestrel yielded an FDR of 0.01, but a sensitivity of only 0.71. GATK yielded an FDR of 0.00 (0.0029) and a sensitivity of 0.84 If we examine only regions where Kestrel had an active region, the sensitivity rose to 0.98 for Kestrel and 0.97 for GATK, and the FDR remained 0.01 for Kestrel and 0.00 for GATK.

Although the sensitivity was lower for Kestrel, there are regions where Kestrel consistently makes calls that are missed by the GATK pipeline. We compared all Kestrel TP calls that were not within 50 bp of any GATK TP call, merged the resulting calls within 50 bp, and found 780 regions affecting 86 unique genes where calls were missed by GATK in more than 20 samples (Supplementary Table S5 and Supplementary Fig. S7).

4 Discussion

The Kestrel algorithm is a framework for calling variants from sequence data using evidence found only in k-mer space. It provides a mechanism for detecting regions of variation, a method for resolving the variation to variant calls, and a set of heuristics that make it work with real data. Inference on k-mers has been applied to RNAseq analysis (Bray *et al.*, 2016; Patro *et al.*, 2014), metagenomics (Wood and Salzberg, 2014), phylogenetics (Gardner and Slezak, 2010; Gardner *et al.*, 2013) and many other problems in bioinformatics. To our knowledge, Kestrel is a first-in-class variant calling implementation using only k-mers.

Although such a method is unorthodox with respect to current standards, it is able to capture variation in regions where alignments are too noisy. In these cases, a k-mer approach can displace alternatives that require far more computing resources and time to complete, such as *de novo* assemblies. By applying this method to other domains where assemblies are the current standard, such as MLST and pathogen surveillance pipelines, this approach can have a dramatic effect on computing resources.

Despite several advantages, a purely k-mer-based algorithm such as this does have limitations. The most significant of these is that paired-end and sequence read context is lost when shredding sequence reads into k-mers. This information is valuable for resolving variants in repetitive, duplicated or low complexity regions of the genome. Some evidence of these events is still present in k-mer space, such as frequency peaks, and although Kestrel attempts to work through these events, alignments or assemblies will often have higher accuracy.

As a practical tool, Kestrel is a fast alternative to existing methods when applied to specific regions of the genome. We expect it to find use as a targeted tool, as part of automated surveillance pipelines or an orthogonal approach for other callers. In its current form, we cannot purport it to be a whole-genome alternative for software like Cortex or Platypus. Future developments may improve the limitations of this approach, or the algorithm may find use in a hybrid tool making use of k-mers and other data. For example, the GATK HaplotypeCaller might fall-back to such a k-mer method to rescue variants in a region where alignments suffer because of dense SNVs or large indels.

In many cases, algorithms that are free of alignments and assemblies can reduce the demand on computing resources. As high-volume sequencing technology becomes faster and cheaper, reducing the cost of data storage and analysis becomes critical (Köser *et al.*, 2012; Sboner *et al.*, 2011). Kestrel's contribution was only marginal for calling variants where alignments would suffice, but its ability to perform analysis without more costly methods gives it a significant advantage. The computing resources Kestrel requires is well within the capabilities of a modern laptop, which is far less expensive than the powerful machines or computing clusters that would be required to do the same work in the same amount of time. Such a cost savings may conserve vital research funds or enable routine analysis where funding and equipment are not as readily available.

Acknowledgements

Ben Metcalf and Bernard Beall from the Centers for Disease Control and Prevention provided data and insight for the *S.pneumoniae* experiment. Dawn Audano designed the graphic for Figure 1.

Funding

This work was supported by the National Institutes of Health. P.A.'s contribution funded through T32GM105490, which was made possible by Dr. Gregory Gibson. F.O.V. and R.S. are supported by R01EB025022, R01ES026243, and funds awarded to F.O.V. by the Georgia Institute of Technology.

Conflict of Interest: none declared.

References

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Audano,P. and Vannberg,F. (2014) KAnalyze: a fast versatile pipelined K-mer toolkit. *Bioinformatics*, **30**, 2070–2072.

Bankevich,A. *et al.* (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.*, **19**, 455–477.

Bray,N.L. *et al.* (2016) Near-optimal probabilistic RNA-seq quantification. *Nat. Biotechnol.*, **34**, 888–525527.

Cleary,J.G. *et al.* (2015) Comparing variant call files for performance benchmarking of next-generation sequencing variant calling pipelines. *bioRxiv*, 023754.

Eddy,S.R. (2004) What is dynamic programming? *Nat. Biotechnol.*, **22**, 909–910.

Gardner,S.N. *et al.* (2013) When whole-genome alignments just won't work: kSNP v2 software for alignment-free SNP discovery and phylogenetics of hundreds of microbial genomes. *PLoS One*, **8**, e81760.

Gardner,S.N. and Slezak,T. (2010) Scalable SNP Analyses of 100+ Bacterial or Viral Genomes. *J. Forensic Res.*, **1**, 107.

Huang,W. *et al.* (2012) ART: A next-generation sequencing read simulator. *Bioinformatics*, **28**, 593–594.

Iqbal,Z. *et al.* (2012) De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat. Genet.*, **44**, 226–232.

Jolley,K.A. and Maiden,M.C.J. (2010) BIGSdb: Scalable analysis of bacterial genome variation at the population level. *BMC Bioinformatics*, **11**, 595.

Köser,C.U. *et al.* (2012) Routine use of microbial whole genome sequencing in diagnostic and public health microbiology. *PLoS Pathogens*, **8**, e1002824.

Laible,G. *et al.* (1991) Interspecies recombinational events during the evolution of altered pbp 2x genes in penicillin-resistant clinical isolates of *Streptococcus pneumoniae*. *Mol. Microbiol.*, **5**, 1993–2002.

Larsen,M.V. *et al.* (2012) Multilocus sequence typing of total genome sequenced bacteria. *J. Clin. Microbiol.*, **50**, 1355–1361.

Li,G. *et al.* (2012) Complete genome sequence of *Streptococcus pneumoniae* Strain ST556, a multidrug-resistant isolate from an otitis media patient. *J. Bacteriol.*, **194**, 3294–3295.

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

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

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

Li,Y. *et al.* (2010) State of the art de novo assembly of human genomes from massively parallel sequencing data. *Hum. Genomics*, **4**, 271–277.

Li,Y. *et al.* (2016) Penicillin-binding protein transpeptidase signatures for tracking and predicting β-lactam resistance levels in *Streptococcus pneumoniae*. *mBio*, **7**, 1–9.

Martin,C. *et al.* (1992) Relatedness of penicillin-binding protein 1a genes from different clones of penicillin-resistant *Streptococcus pneumoniae* isolated in South Africa and Spain. *EMBO J.*, **11**, 3831–3836.

McKenna,A. *et al.* (2010) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, **20**, 1297–1303.

Ochman,H. *et al.* (2000) Lateral gene transfer and the nature of bacterial innovation. *Nature*, **405**, 299–304.

Olson,N.D. *et al.* (2015) Best practices for evaluating single nucleotide variant calling methods for microbial genomics. *Front. Genet.*, **6**, 1–15.

Patro,R. *et al.* (2014) Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nat. Biotechnol.*, **32**, 462–464.

Reuter,S. *et al.* (2013) Rapid bacterial whole-genome sequencing to enhance diagnostic and public health microbiology. *JAMA Int. Med.*, **173**, 1397–1404.

Rimmer,A. *et al.* (2014) Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications. *Nat. Genet.*, **46**, 912–918.

Roberts,M. *et al.* (2004) Reducing storage requirements for biological sequence comparison. *Bioinformatics*, **20**, 3363–3369.

Salipante,S.J. *et al.* (2015) Large-scale genomic sequencing of extraintestinal pathogenic *Escherichia coli* strains. *Genome Res.*, **25**, 119–128.

Sboner,A. *et al.* (2011) The real cost of sequencing: higher than you think! *Genome Biol.*, **12**, 125.

Smith,T. and Waterman,M. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Thomsen,M.C.F. *et al.* (2016) A bacterial analysis platform: an integrated system for analysing bacterial whole genome sequencing data for clinical diagnostics and surveillance. *PLoS One*, **11**, 1–14.

Wood,D.E. and Salzberg,S.L. (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.*, **15**, 1–12.