Sequence analysis

Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly

Heng Li
Medical Population Genetics Program, Broad Institute, 7 Cambridge Center, MA 02142, USA

ABSTRACT

Motivation: Eugene Myers in his string graph paper suggested that in a string graph or equivalently a unitig graph, any path spells a valid assembly. As a string/unitig graph also encodes every valid assembly of reads, such a graph, provided that it can be constructed correctly, is in fact a lossless representation of reads. In principle, every analysis based on whole-genome shotgun sequencing (WGS) data, such as SNP and insertion/deletion (INDEL) calling, can also be achieved with unitigs.

Results: To explore the feasibility of using de novo assembly in the context of resequencing, we developed de novo assembler, fermi, that assembles illumina short reads into unitigs while preserving most of information of the input reads. SNPs and INDELS can be called by mapping the unitigs against a reference genome. By applying the method on 35-fold human resequencing data, we showed that in comparison to the standard pipeline, our approach yields similar accuracy for SNP calling and better results for INDEL calling. It has higher sensitivity than other de novo assembly based methods for variant calling. Our work suggests that variant calling with de novo assembly can be a beneficial complement to the standard variant calling pipeline for whole-genome resequencing.

In the methodological aspects, we propose FMD-index for forward-backward extension of DNA sequences, a fast algorithm for finding all super-maximal exact matches and one-pass construction of unitigs from an FMD-index.

Availability: http://github.com/lh3/fermi

Contact: hengli@broadinstitute.org

Received on April 3, 2012; revised on May 2, 2012; accepted on May 3, 2012

1 INTRODUCTION

The rapidly decreasing sequencing cost has enabled whole-genome shotgun (WGS) resequencing at an affordable price. Many software packages have been developed to call variants, including SNPs, short insertions and deletions (INDELS) and structural variations (SVs), from WGS data. At present, the standard approach to variant calling is to map raw sequence reads against a reference genome and then to detect differences from the reference. It is well established and local assembly (Carnevali et al. 2011) and SGA (Simpson and Durbin, 2011) as of now, do not explicitly output heterozygotes. Although in theory it is possible to recover heterozygotes from their intermediate output, it may be difficult in practice as the assemblers may not distinguish heterozygotes from sequencing errors. Cortex (Iqbal et al. 2012) is specifically designed for retaining heterozygous variants in an assembly, but it may be missing heterozygotes. ALLPATHS-LG (Simpson et al. 2009) also paid particular attention to keep heterozygotes, but it still has a relatively low sensitivity. In addition, ALLPATHS-LG only works with reads from libraries with distinct insert size distributions and prefers read pairs with mean insert size below three times of the read length, whereas many resequencing projects do not meet these requirements and thus ALLPATHS-LG may not be applied or work to the best performance. Even if we also include de novo assemblers developed for capillary sequence reads, the version of the Celera assembler used for assembling the HuRef genome (Simpson et al. 2009) is the only one that retains heterozygotes while capable of assembling a mammalian genome. At last, one may think to map sequence reads

1838 © The Author 2012. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oup.com
back to the assembled contigs to recover heterozygous events, but this procedure will be affected by the same flaw of read mapping. To the best of our knowledge, no existing de novo assemblers are able to achieve the sensitivity of the standard mapping-based approach for a diploid mammalian genome.

We will discuss that the assembly based variant calling can achieve an SNP accuracy close to the standard mapping approach and have particular strength in INDEL calling, confirming previous studies (Ihabal et al., 2018). In addition, the de novo assembly algorithm, ferm, developed for this practice is also a capable assembler for human assembly.

2 METHODS

The methods section is organized as follows. We first review the history of de novo assembly in the theoretical aspects, which leads to the rationale behind ferm: to use unitigs as a lossless representation of reads. We then summarize the notations used in the article and introduce bidirectional FM-index for DNA sequences. We will present several algorithms for assembling using the bidirectional FM-index. The key algorithm is based on previous works (Simpson and Durbin, 2010), but we need to adapt it to our new index. We also remove the recursion in the original algorithm. Finally we will discuss practical concerns in implementation.

2.1 Theoretical background

2.1.1 A history of the OLC paradigm

Computer assisted sequence assembly can be dated back to the late 1970s (Britten and Davidson, 1979). In 1984, Medvedev et al. first formulated the DNA assembling problem as finding the shortest string (the assembly) such that each sequence read can be mapped to the assembly within a required error rate. To solve the problem, they proposed a three-step procedure, which is essentially the overlap-layout-consensus (OLC) approach. Myers (1991) pointed out that reducing DNA assembly to a shortest string problem is flawed in the presence of repeats. He further proposed the concept of overlap graph, where a vertex corresponds to a read and a bidirectional edge to an overlap. Naively, the DNA assembling problem can be cast as finding a path in the overlap graph such that each vertex/reads is visited exactly once (though edge/overlap caused by repeats are not required to be traversed), equivalent to a Hamilton path problem which is known to be NP-complete. This has led many to believe that the OLC approach is theoretically crippled.

However, it is worth pointing out that although the assembly problem can be reduced to a Hamilton path problem, it can be reduced to other problems as well and in practice almost no assemblers try to solve a Hamilton path problem. We note that a fundamental difference between a genetic graph and an overlap graph is the latter can be trivially reduced while retaining the read relationship. More formally, if \( v_1 \rightarrow v_2 \), \( v_2 \rightarrow v_3 \) and \( v_1 \rightarrow v_3 \) are all present, edge \( v_1 \rightarrow v_2 \) is said to be reducible. When we removed all the contained reads and reducible edges, a procedure called transitive reduction, the resulting graph is still a loyal representation of the overlap graph (Medvedev et al., 2009), but the path corresponding to the assembly is not a Hamilton path anymore because reads from repetitive regions need to be traversed multiple times.

In a transitively reduced graph, if there exists \( v_1 \rightarrow v_2 \) with the out-degree of \( v_1 \) and in-degree of \( v_2 \) both equal to 1, we are able to merge \( v_1 \) and \( v_2 \) into one vertex without altering the topology of the graph. After we performed all possible merges, we get a unitig graph in which each vertex corresponds to a unitig, representing a maximal linear sequence that can be resolved by reads. Multiple copies of a repeat may be collapsed to a single unitig. The concept of unitig helps to greatly simplify an assembly graph. It has played behind ferm: to use unitigs as a lossless representation of reads. We then will discuss practical concerns in implementation.

2.2 Rationale

Being coherent, a perfectly constructed unitig graph annotated with per-unitig read counts in fact encapsulates all the information of reads and encodes no information invalidated by reads. In this sense, any unitig-based analysis has an equivalent read-based analysis, and vice versa. This article just uses this property to explore the applications for which we usually rely on reads.
Table 1. Notations

| Symbol | Description |
|--------|-------------|
| $T$    | String $T=a_1a_2\ldots a_n$, with $a_i\in \Sigma$ |
| $|T|$    | Length of $T$ including terminals: $|T|=n$ |
| $T(i)$ | The $i$-th symbol in string $T$; $T(i)=a_i$ |
| $T(i,j)$ | Substring $T[i,j]=a_ia_{i+1}\ldots a_j$ |
| $S$    | Suffix array $S(i)$ is the position of the $i$-th smallest suffix in $T$ |
| $B$    | BWT: $B[i]=T(S(i)-1)$ if $S(i)<n$ and $B[i]=n$ otherwise |
| $C(a)$ | Accumulated count array: $C(a)=|\{0\leq i<n:T(i)=a\}|$ |
| $O(a,k)$ | Occurrence array: $O(a,k)=|\{0\leq i<n:B[i]=a\}|$ |
| $P$    | String concatenation of string $P$ and $W$ |
| $PW$   | String concatenation of string $P$ and symbol $a$: $PW=P\circ a$ |
| $\mathcal{P}$ | Watson-Crick reverse complement of DNA string $P$ |

2.3 Strings and FM-index

2.3.1 Strings with multiple sentinels. Let $\Sigma=\{\Sigma_1,\Sigma_2,\Sigma_3,\Sigma_4\}$ be the alphabet of DNA sequences with a predefined lexicographical order $\Sigma_1<\Sigma_2<\Sigma_3<\Sigma_4$, where ‘$\Sigma_1$’ represents an ambiguous base and ‘$\Sigma_3$’ is a sentinel that marks the end of a string. An element in $\Sigma$ is called a symbol and a sequence of symbols is called a string. Given a string $T$, let $T[i]$ be the length of the string, $T(i)$, $i=0,\ldots,|T|-1$, be the $i$-th symbol in the string $T$, $T[i]=a_i$; $0\leq i<|T|$ be a substring and $T[i,|T|]=T[i]$ be a suffix of $T$ (Table 1). Following the definition by Siren (2009), we define a string terminated with ‘$\Sigma_3$’ as a text. A test may have multiple sentinels. In a test $T$, if $T[1]=5$ and $T[|T|]=5$, we mandate $T[1]=T[|T|]$ if and only if $i=j$. Thus when we compare two strings $P$ and $W$, we do not need to compare beyond a sentinel because each sentinel has a different lexicographical rank.

For two strings $P$ and $W$, let $P\circ W$ be their string concatenation. We may sometimes write $P\circ W$ as $PW$ if it is unambiguous in the context. Given an ordered set of texts, we call their ordered string concatenation as a collection, which is also a test. For example, suppose we have two reads. The first is \texttt{GTG} and the second is \texttt{3'}. For convenience, we assign an integer from 0 to 5 to ‘$\Sigma_1$’ and ‘$\Sigma_2$’.

For two strings $P$ and $W$, let $P\circ W$ be their string concatenation. We may sometimes write $P\circ W$ as $PW$ if it is unambiguous in the context. Given an ordered set of texts, we call their ordered string concatenation as a collection, which is also a test. For example, suppose we have two reads. The first is \texttt{GTG} and the second is \texttt{3'}. For convenience, we assign an integer from 0 to 5 to ‘$\Sigma_1$’ and ‘$\Sigma_2$’.

2.3.2 FM-index. The suffix array $S$ of a text $T$ is a permutation of integers between 0 and $|T|-1$, where $S(i)$, $0\leq i<|T|$, is the position of the $i$-th smallest suffix of $T$. Given a string $P$, the suffix array interval $I(P)$ of $P$ in $T$ is defined as

$$I(P)=\max[k:P'[k]\circ P\circ P[P]]=\min[k:P'[k][P')(P)]$$

For convenience, we also define $I(P')=I(P)=I(P')\circ I(P)+1$ as the size of the interval.

The Burrows-Wheeler Transform (BWT) or BWT, is a permutation of symbols in $T$. The BWT string $B$ is computed as $B[i]=T(S[i]-1)$ for $S[i]<n$ and $B[i]=n$ otherwise. Given a test $T$, we may compute the cumulative count array $C(a)$ as the number of symbols in $T$ that are lexicographically smaller than $a$, and the occurrence array $O(a,k)$ as the occurrence of symbols $a$ in $B[0:k]$.

An FM-index (Ferragina and Manzini, 2000) is a compressed representation of the BWT $B$, the occurrence array $O(a,k)$ and the suffix array $S(i)$. The key property of FM-index is

$$I'(aP)=C(a)+O(a,I'(P)-1)$$

and $I'(aP)=C(a)+O(a,I'(P)+1)$ if and only if $aP$ is a substring of $T$. We note that these two equations are different from the ones in our previous paper (Li et al., 2011). Durbano (2009) defined that $C(a)$ and $O(a)$ defined here include the sentinels, but the two arrays in the previous paper exclude them.

Given a collection $T=O(Q_1, Q_2)$, we can retrieve sequence $Q_1$ in linear time with Algorithm 1 (Durr et al., 2002). The second return value is the rank of $Q_1$, which equals $|Q_1|\circ Q_2$.

Algorithm 1. Sequence retrieval

Input: Sequence index $i\geq 0$;
Output: Sequence $P$ and $k$, the rank of $P$

Function GetSeq($i$) begin

$k\leftarrow 1$

$P\leftarrow$ empty string;

while true do

$a\leftarrow B[k]$;

if $a=C(0)+O(a,k)-1$ then

\[P\leftarrow aP\]

$k\leftarrow k+1$

end)

end

2.4 FMD-index

Given DNA texts $R_1,\ldots, R_m$, define $T=R_1R_2R_3\ldots R_m$ as the bidirectional collection of the texts. We call the FM-index of $T$ as the FMD-index of $R_1,\ldots, R_m$ and define the bi-interval of a string $P$ as $\langle I(P), I'(P), I'(P')\rangle$. We will show how to compute the bi-interval of $aP$ and $Pa$ when we know the bi-interval of $P$.

We note that when we know the bi-interval of $P$, $I'(aP)$ and $I'(P')$ can be readily computed with Equation 1. $\langle I'(aP), I'(P), I'(P')\rangle$ is a sub-interval of $\langle I'(P), I'(P'), I'(P')\rangle$ because $P'$ is a prefix of $P'\circ P'\circ P$. Due to the innate symmetry of $T$, $I'(P)=I'(P')$ for all $a\in \Sigma$. We can compute $I'(P')$ for all $a\in \Sigma$ with Equation 2. use these interval sizes to divide $\langle I'(P), I'(P'), I'(P')\rangle$ and finally derive $\langle I'(aP), I'(P), I'(P')\rangle$. This completes the computation of the bi-interval of $aP$ (Algorithm 2). Furthermore, when we backward extend $P$, we actually forward extend $P$. Conversely, backward extension of $P'$ yields forward extension of $P$ (Algorithm 3). An FMD-index is bidirectional.

In comparison to the bidirectional BWT, FMD-index builds both forward and reverse strand DNA sequences in one index. Although the FMD-index is not applicable to generic texts, it is conceptually more consistent with double-strand DNA and improves the speed of exact matching as we only need to search against one index. For example, BWA-SW Li and Durbin (2010) gets a 80% speedup when we adopt the FMD-index as the data structure.

2.5 Unigal construction

2.5.1 Labeling reads and overlaps. Given a bidirectional collection $T=R_1R_2\ldots R_m$, label the $i$-th input read $R_i$ with an ordered integer pair $[k, l]$, where $k$ is the rank of $R_i$ and $l$ the rank of $R_i$. The pair $[k, l]$ can be computed by GetSeq(2) and GetSeq(2i-1), respectively. Note that, if read $R_i$ is labeled by $[k, l]$, read $R_i$ should be labeled by $[l, k]$, with the two integer swapped.

For two reads labeled by $[k, l]$ and $[k', l']$, if the tail (3' end) of read $[k, l]$ overlaps the head (5’ end) of $[k', l']$, we use an unordered integer pair $(k, l)$ to label the overlap. Such a tail-to-head overlap. Similarly, we use
An FMD-index can be used to find supermaximal exact matches (SMEs) between a reference and a query sequence. Formally, a maximal exact match (MEM) is an exact match that cannot be extended in either direction of the match. An SME is a MEM that is not contained in other MEMs on the query sequence. Fermi uses SMEs to map reads back to the unitigs.

Algorithm 2: Backward extension
Input: Bi-interval \([k, l, r]\) of string \(W\) and a symbol \(a\)
Output: Bi-interval of string \(aW\)

Function BACKWARDEXT([k, l, r], a) begin
for \(b = 0\) to 5 do
\(k_b = C(b) + O(b, k - 1)\)
\(l_b = C(b) + O(b, k + l - 1)\)
\(r_b = C(b) + O(b, k + r - 1)\)
for \(b = 0\) to 3 do
\(l_b = l_b + l_b + 1\)
\(r_b = r_b + r_b + 1\)
return \([k_b, l_b, r_b]\)
end

Algorithm 3: Forward extension
Input: Bi-interval \([k, l, r]\) of string \(W\) and a symbol \(a\)
Output: Bi-interval of string \(Wa\)

Function FORWARDEXT([k, l, r], a) begin
\([l', k', r'] = \text{BACKWARDEXT}([k, l, r], a)\)
return \([k', l', r']\)
end

2.5.2 Finding irreducible overlaps
Finding irreducible overlaps plays a central role in fermi as well as in SGA. Given their importance, we present a restructured version of this algorithm (SD10; Simpson and Durbin, 2011) using our notations (Algorithm 4).

In Algorithm 4, Line 1 computes the bi-interval of a single symbol. The loop at Line 2 uses backward extensions to find all the reads overlapping with the input string \(P\). The loop at Line 3 uses forward extensions base by base to exclude reducible overlaps found at the previous step. \(W\) is this loop keeps the common substring of reads overlapping \(P\) extended from the 3’ end of \(P\). If in an iteration we find the sentinel of a read \(R\) (Line 5), then all the reads sharing the same \(W\) with \(R\) must overlap with both \(R\) and \(P\) and therefore their overlaps with \(P\) are reducible. In this case, no further forward extensions are necessary (Lines 4 and 6).

Similar to the original algorithm, Algorithm 4 requires that there are no contained reads. Fermi actually implements a modified version that detects reads containment on the fly, but we think the algorithm is a little overcomplicated. It is probably easier to filter contained reads first and then run Algorithm 4, as SGA does.

2.5.3 Unitig construction
Unitig construction is a process of unambiguous merge of overlapped reads. If \([k, l]\) and \([k', l']\) have an irreducible overlap \((k, k')\) and can be unambiguously merged, we label the merged sequence with \([k, j]\), the similar can be applied to other three types of overlaps. With this simple labeling procedure, we are able to fully keep track of the graph topology during the unitig construction and without staging the graph in RAM. This procedure can also be easily multi-threaded.

2.6 Finding the SMEs
An FMD-index can be used to find supermaximal exact matches (SMEs) between a reference and a query sequence. Formally, a maximal exact match (MEM) is an exact match that cannot be extended in either direction of the match. An SME is a MEM that is not contained in other MEMs on the query sequence. Fermi uses SMEs to map reads back to the unitigs.

Algorithm 4: Finding irreducible overlaps
Input: Read \(P\) and the minimum overlap length \(x\)
Output: Set of bi-intervals of reads having irreducible overlaps with the 3’ end of \(P\)

Function IRROVERLAP(P, s) begin
Initialize \(\text{Curr}\) and \(\text{Prev}\) as empty arrays;
\(a = P[|P| - 1]\);
\([k, i, a] = (C(a), C(a), C(a + 1) - C(a));\)
for \(i = |P| - 2\) to 0 do
if \(P[i] = a\) then
\([k', i', a'] = \text{BACKWARDEXT}([k, i, a], 0)\);
if \(i' \neq 0\) then
\(\text{Append} ([k', i', a'], \text{Curr});\)
\([k, i, a] = \text{BACKWARDEXT}([k, i, a], P[i])\);
end
end
Reverse \(\text{Curr}\) and swap \(\text{Curr}\) and \(\text{Prev}\);
\(\text{Finished} = \emptyset;\)
end
while \(\text{Prev}\) is not empty do
Reset \(\text{Curr}\) to empty;
for \([k, i, a] \in \text{Prev}\) do
if \(W \in \text{Finished}\) then
continue;
\([k', i', a'] = \text{FORWARDEXT}([k, i, a], 0)\);
if \(a' \neq 0\) then
\(\text{Finished} = \text{Finished} \cup [W];\)
end
\(I = I \cup ([k', i', a'])\);
end
end
for \(a = 1\) to 5 do
\([k', i', a'] = \text{FORWARDEXT}([k, i, a], a);\)
if \(i' \neq 0\) and \([k', i', a']\) is not in \(\text{Curr}\) then
\(\text{Append} ([k', i', a'], \text{Curr});\)
end
end
return \(\text{IRR} \cup \text{Ovlp}\)
end

2.7.3 Simplifying complex bubbles

A bubble is a directed acyclic graph with a single source and a single sink having at least two paths between the source and the sink. A closed bubble is a bubble with no incoming edges from or outgoing edges to other parts of the entire graph, except at the source and the sink vertices. A closed bubble is simple if there are exactly two paths between the source and the sink; otherwise it is complex.

In de novo assembly, a bubble is frequently caused by sequencing errors or heterozygotes. Most short-read assemblers uses a modified Dijkstra’s algorithm to pop bubbles progressively. Such an algorithm works fine for haploid genomes, but it is not straightforward to distinguish heterozygotes from errors when the bubble is complex.

Fermi uses a different algorithm. It effectively performs topological sorting from the end of a vertex while keeping track of the top two paths containing most reads. A bubble is detected when every path ends at a single vertex. It then drops vertices not on the top two paths and thus turns a complex bubble to a simple one.

2.7.4 Using the paired-end information

Given paired-end reads with short-insert sizes, fermi maps reads back to the unitigs with Algorithm 5. If two unitigs are linked by at least five read pairs, fermi will locally assemble the ends of unitigs together with unpaired reads pointing to the gap under a relax setting. Fermi tries to align the ends of unitigs using the Smith–Waterman algorithm, which may reveal imperfect overlaps caused by sequencing errors or heterozygotes. Fermi also uses paired-end reads to break contigs at regions without bridging read pairs. This helps to reduce misassemblies during the unitig construction.

3 RESULTS

We evaluated fermi on 101 bp paired-end reads from NA12878 [Buenrostro et al. 2011]. The total coverage of the original data is ∼70-fold, but we only used half of them. We assembled the 35-fold reads with fermi on a machine with 12 CPUs and 96 GB memory in ∼5 days. The peak memory usage is 92 GB.

We obtained unitigs of N50 1022 bp, totaling 3.83 GB. After collapsing most heterozygotes and closing gaps with paired-end reads, we got longer contigs (Table 4). Unitigs are short and redundant mainly because they break at heterozygotes.

For SNP and INDEL calling, we aligned unitigs to the reference genome using BWA-SW [Li and Durbin 2009] with command line options ‘-b9 -q16 -r1 -w500’. We called SNPs with the SAMtools caller and called INDELs by directly counting INDELs from the pileup output. We did not run a standard INDEL caller as short-read INDEL callers do not work well with long contig sequences.

3.1 Performance on de novo assembly

We obtained the HuRef capillary read assembly [Keye et al. 2011] and the ALLPATHS-LG NA12878 contigs (AC:AEKP01000000) from NCBI, the SGA scaffolds from https://bit.ly/jts12878 (Simpson and Durbin 2012) and the ABySS assembly provided by Shaun Jackman (personal communication). For both SGA and ABySS scaffolds, we split at any ambiguous bases to get contigs; for the HaRef assembly, we split at contiguous ‘N’ longer than 20 bp. The ABySS, fermi and SGA scaffolds are derived from essentially the same input reads. ALLPATH-LG uses a superset of reads at 100-fold coverage, including reads from multiple long-insert libraries.

From Table 2, we can see that the HuRef assembly has much better contiguity than short-read assemblies. It appears to yield more alignment break points, some of which may be caused by true SVs not easily detectable with short reads. The quality of short-read assemblies varies in terms of contiguity, misassembly rate and redundancy between contigs, but overall, they are largely comparable to each other.

3.2 Performance on SNP and INDEL calling

One of the key motivations of fermi is to explore the power of de novo assembly in calling short variants. We collected several SNP and INDEL call sets (Table 3) and compared the performance of fermi (Tables 4 and 5).
Table 2. Statistics on human whole-genome assemblies

| Label | Data            | Assembler | Mapper | Caller |
|-------|-----------------|-----------|--------|--------|
| AC    | 96X Illumina PE  | AllPaths-LG | BWA-SW | SAMtools |
| BS    | 70X Illumina PE  | BWA1      | SAMtools |
| CG    | Complete Genoms | cgatools2  | cgatools2 |
| CV    | 26X Illumina SE  | Cortex     | Cortex-var |
| FC    | 35X Illumina SE  | Fermi      | BWA-SW | SAMtools |
| MD    | 60X multiple    | MAQ        | 1000 g pilot |
| MI    | Capillary reads |           |        |
| SS    | 35X Illumina SE  | BWA-SW     | SAMtools |

a AS uses reads from Illumina jumping and fungal libraries.
b BWA-SW is invoked with 'bwa bwasw -b9 -q16 -r1 -w500'.
c INDELS are called from alignment using the SAMtools caller.
d Realigned by GATK. e [g] contains only Q15/HIGH calls retained.
e CV, FC and SS do not use the pairing information in calling.
f 1000 Genomes Project pilot calls generated from Dindel and multiple SNP callers.
g INDELs in call set h is found in 'ALL' if it is found in one of the other INDEL sets in the table, plus the AC call set. In the table, a number on the diagonal equals \(|R\) and the number of INDEL calls in call set \(g\) is found in \(C\).

Table 3. Evaluated SNP and INDEL call sets

| Label  | Data            | Assembler | Mapper | Caller |
|--------|-----------------|-----------|--------|--------|
| FC     | CV              | SS        | BS     | CG     | MD     |
| No. of SNPs (M) | 3.37 | 2.20 | 3.24 | 3.50 | 3.34 | 2.69 |
| No. of hits (M)  | 1.97 | 1.04 | 1.94 | 2.11 | 2.04 | 1.65 |
| Ts/Tv  | 2.04 | 2.03 | 2.08 | 2.11 | 2.12 | 2.06 |
| DN50 (bp) | 3593 | 6662 | 3523 | 3392 | 3447 | 3992 |
| DN2/DN50 | 22.3 | 20.8 | 23.4 | 22.7 | 22.3 | 22.9 |

Ts/Tv is the transition-to-transversion ratio of SNPs. DN50 is calculated as follows. The reference genome is masked according to the alignability mask (http://blat)y and segmented into intervals at heterozygous SNPs. DN50 is computed such that 95% of unique positions in the genome are in intervals longer than DN50. DN2 is calculated similarly and DN2/DN50 is the ratio of DN2 and DN50. DN50 measures the sensitivity; the smaller the better. DN2/DN50 measures the precision of heterozygous SNPs; the higher the better.

For SNP calling (Table 4), fermi misses 3% of SNPs called in SS, but finds more additional ones. Manual examination reveals that the additional calls are mainly caused by two factors. Firstly, in the single-end mode, BWA-SW is very conservative. It may consistently give a correct alignment a low-mapping quality which can be patched with gaps no longer than 500 bp.

4 DISCUSSIONS

In this article, we derived FMD-index by storing both forward and reverse complement DNA sequences in FM-index. This simple modification enables faster forward–backward search than

Table 4. Statistics of SNP call sets

| MD   | CG   | BS   | CV   | FC   | MI   | ALL |
|------|------|------|------|------|------|-----|
| No. of SNPs (M) | 3.37 | 2.20 | 3.24 | 3.50 | 3.34 | 2.69 |
| No. of hits (M)  | 1.97 | 1.04 | 1.94 | 2.11 | 2.04 | 1.65 |
| Ts/Tv  | 2.04 | 2.03 | 2.08 | 2.11 | 2.12 | 2.06 |
| DN50 (bp) | 3593 | 6662 | 3523 | 3392 | 3447 | 3992 |
| DN2/DN50 | 22.3 | 20.8 | 23.4 | 22.7 | 22.3 | 22.9 |

TS/tv is the transition-to-transversion ratio of SNPs. DN50 is calculated as follows. The reads into longer sequences which increase the power of BWA-SW.

Secondly, in the fermi alignment, some regions may be mapped with a high-mismatching rate. These may be due to small-scale misalignments in fermi utigs or in the reference assembly, or copy-number variations. It is possible that these clustered SNPs contain more errors. Such errors may lead to reduced ts/tv, but tend not to break long homozygous blocks due to very recent coalescences.

In this article, we derived FMD-index by storing both forward and reverse complement DNA sequences in FM-index. This simple modification enables faster forward–backward search than

Table 5. Fraction of INDELs found in other call sets

| MD   | CG   | BS   | CV   | FC   | MI   |
|------|------|------|------|------|------|
| No. of type-2 breaks | 1617 | 3823 | 1120 | 1735 | 6626 |
| No. of type-1 breaks | 5856 | 13738 | 5704 | 6049 | 16318 |
| Contigs over 150 bp in length are aligned to the human reference genome GRCh37 with BWA-SW using option "-b33 -q50 -e151". A type-1 break point is detected if a contig is split during alignment and mapped to two distinct locations, and at each location the alignment is longer than 500 bp and the mapping quality is no less than 10. Type-2 break points exclude type-1 break points which can be patched with gaps no longer than 500 bp.

We manually checked 30 missing INDELs in an alignment viewer.

Secondly, in the fermi alignment, some regions may be mapped with a high-mismatching rate. These may be due to small-scale misalignments in fermi utigs or in the reference assembly, or copy-number variations. It is possible that these clustered SNPs contain more errors. Such errors may lead to reduced ts/tv, but tend not to break long homozygous blocks due to very recent coalescences.

In this article, we derived FMD-index by storing both forward and reverse complement DNA sequences in FM-index. This simple modification enables faster forward–backward search than
bi-directional BWT [lam et al. 2004] and makes FMD-index a more natural representation of DNA sequences. Based on FMD-index, we developed a new de novo assembler, ferrmi, which achieves similar quality to other mainstream assemblers.

We demonstrated that it is possible to call SNPs and short INDELs by aligning assembled units to the reference genome. This approach has similar SNP accuracy to the standard mapping-based SNP calling and arguably outperforms the existing methods on INDEL calling in terms of both sensitivity and precision. Assembly based variant calling is a practical and beneficial complement to mapping-based calling.

In the course of evaluating INDEL accuracy, we found that outside long homopolymer regions, INDEL call sets do not often contain false positives, but they may have high false-negative rate, which leads to the apparent small overlap between call sets [lam et al. 2013] as a theoretical result. We note that with read counts kept, units are a lossless but reduced representation of sequence reads. They are 'reduced' in that individual reads are lost; they are ‘lossless’ in that all the information in reads, such as small variants, copy numbers and structural changes are fully preserved in units, as long as they are constructed correctly. For single-end reads, it is theoretically possible to ‘compress’ reads to units, which are largely non-redundant and much smaller in size. Accurately and efficiently constructing units might provide an interesting alternative to data storage and downstream analyses in future, though practical challenges, such as the high-computational cost and the lack of accuracy of units, remain at present.

ACKNOWLEDGEMENTS

We are grateful to David Reich, Evan Echler and Peter Sudmant for providing additional data and computing resources for evaluating ferrmi, to David Altshuler and the GSA group at Broad for the helpful discussions and to Richard Durbin and Jared Simpson for their comments on the initial draft of the manuscript. We also thank Shaun Jackman for providing the ABySS assembly and Michael Brudno for his insights into the algorithm time complexity.

Funding: NIH Joint SNP and CNV calling in 1000 Genomes sequence data grant (U10HG005208-01).

Conflict of Interest: none declared.

REFERENCES

Alten, C.A. et al. (2010) Dindel: accurate indel calls from short-read data. Genome Res., 20, 961–973.
Burrows,M. and Wheeler,D.J. (1994) A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, Palo Alto, CA.
Carnes, E.L. et al. (2011) Computational techniques for human genome resequencing using paired-end reads. J. Comput. Biol., 19, 279–292.
Chaisson, M.J. et al. (2009) De novo fragment assembly with short mate-paired reads: Does the read length matter? Genome Res., 19, 336–346.
Depristo,M.A. et al. (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nat. Genet., 43, 491–498.
Dwamena, R. et al. (2010) Human genome sequencing using unchained base reads on self-assembling DNA nanorays. Science, 327, 78–81.
Elisa P. (1975) Universal codeword sets and representations of the integers. IEEE Trans. Inf. Theory, 21, 194–203.
Ferragina, P. and Manzini, G. (2000) Opportunistic data structures with applications. In FOCS, Redondo Beach, California, USA, IEEE Computer Society, pp. 390–398.
Frigenti, P. et al. (2010) Lightweight data indexing and compression in external memory. In López-Ortiz, A. (ed.), LATIN, Oaxaca, Mexico, volume 6034 of Lecture Notes in Computer Science, Springer, pp. 677–710.
Gingerman, T. B. et al. (1970) Computer programs for the assembly of DNA sequences. Nucleic Acids Res., 7, 529–545.
Gnerre, S. et al. (2011) High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Proc. Natl. Acad. Sci. USA, 108, 1513–1518.
Homer,N. and Nelson,S.F. (2010) Improved variant discovery through local re-alignment of short-read next-generation sequencing data using SRMA. Genome Biol., 11, R9.
Hon,W.K. et al. (2007) A space and time efficient algorithm for constructing compressed suffix arrays. Algorithmica, 48, 23–36.
Idury,R.M. and Waterman,M.S. (1995) A new algorithm for DNA sequence assembly. J. Comput. Biol., 2, 291–306.
Ikinc,i et al. (2011) HGETC accurate error correction in high-throughput sequencing data. Bioinformatics, 27, 295–302.
Ingétal,Z. et al. (2012) De novo assembly and genotyping of variants using colored de bruijn graphs. Nat. Genet., 44, 226–232.
Lam,T.W. et al. (2009) High throughput short read alignment via bi-directional BWT. In USENIX, Washington, DC, USA, pp. 31–36.
Lam,H.Y.K. et al. (2012) Performance comparison of whole-genome sequencing platforms. Nat. Biotechnol., 30, 78–82.
Levy,S. et al. (2007) The diploid genome sequence of an individual human. PLoS Biol., 5, e254.
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. (2011) Improving SNP discovery by base-alignment quality. Bioinformatics, 27, 1157–1158.
Li et al. (2010) De novo assembly of human genomes with massively parallel short read sequencing. Genome Res., 20, 265–272.
Miketin,V. et al. (2009) Storage and retrieval of individual genomes. In Batzoglou,S. (ed.), RECOMB Tucson, AZ, USA, volume 5541 of Lecture Notes in Computer Science, Springer, pp. 121–137.
Munkel,H.M. and Kwiatkowskik,D.P. (2009) SNP-o-matic. Bioinformatics, 25, 2434–2435.
Medvedev,P. et al. (2007) Computability of models for sequence assembly. In Giancarlo,R. and Hannenhalli,S. (eds.), WABI, Philadelphia, PA, USA, volume 4654 of Lecture Notes in Computer Science, Springer, pp. 289–301.
Mills,R.E. et al. (2011) Natural genetic variation caused by small insertions and deletions in the human genome. Genome Res., 21, 830–839.
Myers,E.W. (1995) Toward simplifying and accurately formulating fragment assembly. J. Comput. Biol., 2, 275–280.
Myers,E.W. et al. (2000) A whole-genome assembly of drosophila. Science, 287, 2196–2204.
Myers,E.W. (2005) The fragment assembly string graph. Bioinformatics, 21 (Suppl. 2), i79–i85.
Nong,G. et al. (2011) Two efficient algorithms for linear time suffix array construction. IEEE Trans. Comput., 60, 1471–1484.
Oswalaki,S. et al. (2008) Sequencing of natural strains of arabaicis thaliana with short reads. Genome Res., 18, 2024–2033.
Pelski,J. et al. (1984) SEQAID: a DNA sequence assembling program based on a mathematical model. Nucleic Acids Res., 12, 307–322.
Pevzner,P.A. et al. (2001) An Eulerian path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. USA, 98, 9748–9753.
Simpson,T.J. and Durbin,R. (2010) Efficient construction of an assembly string graph using the FM-index. Bioinformatics, 26, 1367–1373.
Simpson,T.J. and Durbin,R. (2012) Efficient de novo assembly of large genomes using compressed data structures. Genome Res., 22, 549–556.
Simpson,T.J. et al. (2009) ABySS: a parallel assembler for short read sequence data. Genome Res., 19, 1117–1123.
Sier,J. (2009) Compressed suffix arrays for massive data. In String Processing and Information Retrieval, Saarbrücken, Finland, pp. 63–74.
Staden,R. (1995) A strategy of DNA sequencing employing computer programs. Nucleic Acids Res., 23, 2611–2613.
Zerbino,D.R. et al. (2009) Pebbile and rock band: heuristic resolution of repeats and scaffolding in the velvet short-read de novo assembler. PLoS ONE, 4, e4707.
1000 Genomes Project Consortium. (2010) A map of human genome variation from population-scale sequencing. Nature, 467, 1061–1073.