Published online January 26, 2004

562–569 *Nucleic Acids Research, 2004, Vol. 32, No. 2*
DOI: 10.1093/nar/gkh216

Automated correction of genome sequence errors

Pawel Gajer*, Michael Schatz and Steven L. Salzberg

The Institute for Genomic Research, 9712 Medical Center Drive, Rockville, MD 20850, USA

Received November 7, 2003; Revised and Accepted December 16, 2003

ABSTRACT

By using information from an assembly of a genome, a new program called AutoEditor significantly improves base calling accuracy over that achieved by previous algorithms. This in turn improves the overall accuracy of genome sequences and facilitates the use of these sequences for polymorphism discovery. We describe the algorithm and its application in a large set of recent genome sequencing projects. The number of erroneous base calls in these projects was reduced by 80%. In an analysis of over one million corrections, we found that AutoEditor made just one error per 8828 corrections. By substantially increasing the accuracy of base calling, AutoEditor can dramatically accelerate the process of finishing genomes, which involves closing all gaps and ensuring minimum quality standards for the final sequence. It also greatly improves our ability to discover single nucleotide polymorphisms (SNPs) between closely related strains and isolates of the same species.

INTRODUCTION

Large-scale genome sequencing has progressed rapidly in recent years, with advances in sequencing technology leading to a wealth of new genomes for the scientific community. Software for assembling and analyzing genomes has improved dramatically, allowing genome scientists to tackle large, mammalian genomes using the whole genome shotgun (WGS) method, which is much faster and cheaper than earlier approaches.

All sequencing projects depend critically on a base caller, a program that turns the fluorescent signal intensities detected by an automated sequencing machine into a sequence of the four bases of DNA. Base calling software also assigns a probability of error to each base. Since the advent of the Phred program (1,2), virtually all genome sequencing projects have come to rely on these probabilities to help extract the maximum amount of information possible from each sequence. Improvements in capillary sequencing technology have allowed some laboratories to routinely acquire 800 bases of high quality sequence from each sequencing reaction (or 'read').

At present, the rate of genome sequencing far surpasses the rate of finishing genomes. Finishing refers to the process of filling in all the gaps in the initial assembly of a genome and of providing guarantees about the minimum quality of the overall sequence. A WGS project typically results in many islands of overlapping reads called contigs, which are generated by a genome assembler from the collection of sequence reads. The finishing process attempts to turn these contigs into a complete, accurate representation of an organism's chromosomes. Depending on the genome size and the amount of sequence generated, an assembler may produce anywhere from one to many thousands of contigs, each representing two or more reads; only very rarely does it produce contigs representing entire chromosomes and even those are usually very small chromosomes, such as bacterial plasmids. The finishing process requires laboratory scientists to run additional reactions to fill in the gaps in the assembly. It may also involve extensive manual curation of contigs that appear to overlap but that were not correctly assembled together. Sequence finishers often edit the reads comprising these contigs to create a better assembly.

Our experience with dozens of genome projects led us to realize that we could automate much of the finishing process, greatly accelerating it and at the same time producing a far higher quality genome assembly. To accomplish this goal, we have developed a new type of base caller, one that uses the assembled sequences to guide the algorithm. A critical idea behind our new algorithm, called AutoEditor, is that WGS projects typically produce 6-fold (also called 6X) or higher coverage of a genome, i.e. each base in a genome is covered, on average, by six distinct reads. A genome assembler produces a multiple alignment showing how all the reads map to each contig. Thus for any base in a 6X assembly, on average there are five other bases that align to the same contig position.

This leads naturally to the following observation: a base calling algorithm does not need to limit its input to the raw data (called a chromatogram or, equivalently, an electropherogram) from a single read. For genomes that have been assembled into contigs, a base caller can use the information from all reads that align to a given position in order to inform its decision. In this way we can discover sequencing errors within a read and re-analyze it in a more robust way. Assembling the corrected reads improves the overall quality of the consensus sequence.

A number of algorithms for correcting errors in shotgun sequences have been described previously, including those of Tammi (3), Kececioglu (4), Pevzner (5) and Batzloglou (6). These earlier methods use the alignment of reads produced by

*To whom correspondence should be addressed. Tel: +1 301 795 7854; Fax: +1 301 795 7208; Email: pgajer@tigr.org an assembler, but none of them uses or re-analyzes the chromatogram data to make the correction. AutoEditor is the first system that genuinely recalls bases using both the assembly and the primary signal from the sequencing machine. In principal, this should lead to dramatically more accurate base calling software. As we show below, our experience bears this out.

Another major task of genome sequencing is ensuring the overall accuracy of the genome. For example, the human genome project sought to attain an overall error rate of less than one error per 10 000 bp. Other projects have striven to achieve similar accuracies, and in many cases have significantly surpassed this rate; for example, in bacterial genome projects, finished sequence has been reported to have error rates approaching 1/100 000 (7,8). For the human genome and other large, diploid genomes, differences between overlapping reads might come from several sources, including: (i) true genetic differences such as single nucleotide polymorphisms (SNPs); (ii) sequencing errors; (iii) cloning errors. Considerable effort has already been devoted to SNP discovery in the human genome, however, this process is made much more difficult by sequencing errors, which occur at a higher rate than SNPs. To illustrate, note that individual sequences, after trimming off the ‘bad’ sequence at the ends, have an error rate of ~1%. The human SNP rate is currently still a topic of much debate, but it has been estimated at ~0.16% (9,10). Thus one might expect roughly six times more sequencing errors than true SNPs in a human genome assembly (depending on, among other things, how many different individuals contributed DNA to the sequencing pool). This makes validation of each SNP much more difficult.

AutoEditor eliminates ~80% of sequencing errors, greatly increasing the signal-to-noise ratio in SNP discovery projects. As we show below, it achieves these results consistently across a wide range of genomes.

MATERIALS AND METHODS

AutoEditor consists of an input parser, a module for aligning reads to the consensus sequence, a module for error correction and a module to output corrected sequences. This section describes the error correction algorithm of AutoEditor.

The input to the error correction module is an alignment of reads to a consensus sequence. A sample of this alignment data is depicted in Figure 1.

```
                        111111111122222222223333333
position:     0123456789012345678901234567890123456
consensus:    ATCGATTGGCGATCAACTTTAC*CCATTACCGGGACT

read[0]:      ATCGATTGGCGATCAACTTTAC*CCATTACCGGGACT
read[1]:                 ATCA*CTTTAC*CCATTACCGGGACT
read[2]:      ATCGATTGGCGATCAACCTTAC*CCATTACCGGGACT
read[3]:      ATCGATTGGCGATCAAC*TTACCCCAT
```

Figure 1. Alignment of four reads with a consensus sequence. Asterisks (*) indicate gaps inserted in the reads in order to align them with the consensus.

The set of base calls aligned to a given position is called a slice of the alignment. For example, the slice at position 17 in Figure 1 is the sequence TTC*. A homogeneous slice is a slice all of whose base calls are the same. If any of the bases disagree within a slice, it is a non-homogeneous slice or a discrepancy slice. In Figure 1 the slices at positions 15, 17 and 22 are non-homogeneous.

If at least half of the elements of a slice agree with one another, we call this set the homogeneous majority part of the slice. In WGS data at a reasonable depth of coverage it is very rare that a slice fails to have a homogeneous majority. (In our data, this accounts for <0.01% of discrepancy slices.) We only attempt error corrections on non-homogeneous slices with a homogeneous majority portion, i.e. if no base accounts for at least half of the slice, then we do not attempt to correct it. In the very rare event where a slice consists of just two bases in a 50/50 split, for example TTTTCCCC, the homogeneous majority part is chosen based on lexicographic order, i.e. A is chosen over C, C over G and G over T. This implies that AutoEditor may fail to correct a few miscalled bases, but it will not make any errors as a result of this lexicographic bias.

If the homogeneous majority portion of a slice consists of gaps, the other elements are called insertion bases or insertion elements. For example, the slice ***C at the consensus sequence position 22 in Figure 1 has a single insertion base, C. If the elements of the homogeneous majority part are not gaps, then any gap in the slice is called a deletion element, and each disagreeing base is called a substitution base or a substitution element. For example, the slice TTC* at position 17 in the assembly fragment from Figure 1 has one deletion element (*) and one substitution base, C.

Before any errors are corrected, AutoEditor recomputes the consensus sequence using simple majority rule (ignoring quality values), using lexicographic order to break ties. After the corrections are made, the consensus sequence is re-called again, using a more sophisticated consensus call function that takes into account quality values of the reads. The output of the consensus caller uses ambiguity codes. The final consensus recall is done to ensure that the output of AutoEditor has the consensus sequence consistent with the one computed by other TIGR tools.

The error correction module of AutoEditor consists of three components for processing insertion, deletion and substitution elements of non-homogeneous slices, respectively. Before we describe these modules we will discuss types of errors or anomalies found on the chromatogram level and then address the issue of selection of an appropriate fragment of a chromatogram to analyze a discrepant base.

In order to illustrate the different types of chromatogram anomalies, let us consider the non-homogeneous slice TTC* at position 17 in Figure 1. The consensus base T associated with this slice is a part of a short poly(T) region within the consensus sequence [...CTTTA...]. Usually, a discrepancy within a single base repeat region is caused either by poor quality in the fluorescent signal or by a shift of the signal in the repeat base channel with respect to the signals in the other channels. (Note that each of the four bases are captured in a different, independent color channel. Thus A residues are green, C residues are blue, G residues are yellow and T residues are red.) These two scenarios are illustrated in the following figures. We will refer to signal shift and poor quality anomalies as ‘shift error’ and ‘unresolved peaks error’ respectively. An unresolved peaks error is characterized by shallow or poorly defined local minima between signal peaks.

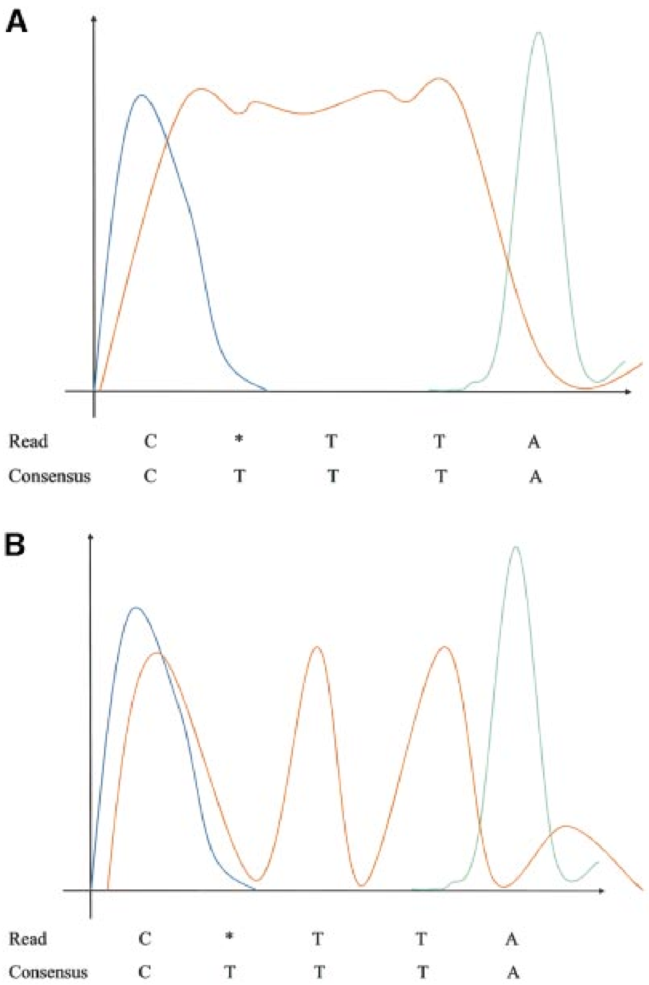

Figure 2. (**A**) Illustrates an unresolved peak error, where the three red T peaks have been interpreted as two broader peaks. (**B**) Shows a signal shift error, where the three red T peaks are clearly resolved, but are shifted left so that the first peak is hidden by the preceding blue C peak. The *x*- and *y*-axes represent chromatogram positions and signal intensities, respectively.

Looking at the examples in Figure 2, we see that analysis of the chromatogram has to be performed in a range that may go well beyond the immediate vicinity of a discrepant base. More precisely, if the discrepancy occurs within a single base repeat region, then one should analyze a region of the chromatogram that contains the entire repeat region together with the flanking bases of the repeat. We will use the phrase 'search region' to refer to the portion of the chromatogram that the algorithm needs to consider in order to analyze a discrepancy slice. For example, in the case of deletion illustrated in Figure 2, the search region would be the fragment of the chromatogram associated with the sequence CTTA.

Computing well-resolved peaks

The heart of the error correction module of AutoEditor is a function that computes the number of well-resolved peaks (captured by the variable *wrp*) in a search region of a chromatogram. We begin with a precise definition of well-resolved peaks and of the function that computes this number.

A peak in a given channel is defined as a fragment of the signal in this channel that contains one local maximum plus its two flanking local minima. With each peak we associate three values: (i) its amplitude, (ii) its 'support' and (iii) the minimum of the differences between its amplitude and its local minima. The amplitude of the peak distinguishes a true peak from noise in the chromatogram. Values (ii) and (iii) determine if the signal is well resolved. The support is intended to capture the width of the peak, but we do not compute this as the distance between the flanking local minima, because of the tendency of peaks to have long tails. Instead, we measure support as the peak width at a certain fraction f of the peak's amplitude (by default $f = 0.25$), illustrated in Figure 3. Note that if the support of two consecutive signal peaks is small, then they must be close to each other. Therefore, support gives us information about the spacing between the peaks.

The number of well-resolved peaks in a given channel within a given search region of a chromatogram is determined as follows. First we compute the median and MAD values of the height and support of *wSize* peaks in the specified channel to the left and to the right of the search region, where *wSize* is a constant (default 15) and MAD is the median absolute deviation of the parameter in question. We take into account only signal peaks for which the base caller assigned a base that agrees with the consensus base. Next, we walk the peaks of the search region, accepting a peak if the following three conditions are satisfied: (i) the height of the peak satisfies $maxVal > \text{median}(height) - hMult \times \text{MAD}(height)$, where *hMult* is a constant (default 3); (ii) the support of the peak is less than or equal to $\text{median}(support) + sMult \times \text{MAD}(support)$, where *sMult* is a constant (default 2); and (iii) the minimum of the differences between the amplitude and local minima of the peak, *relMin*, is greater than a constant (default 50) and $relMin/maxVal > mMult$, where *mMult* is a constant (default 0.2).

The number of well-resolved peaks is equal to the number of accepted peaks within the search region.

The statistics of the amplitude and support of the signal are computed only in the vicinity of the search region because of the tendency for peaks to vary in intensity and resolution over the range of the chromatogram (see Fig. 4) and the tendency for the channels to vary in noise. Thus a peak considered at a given position in a given channel may be acceptable, but that same peak considered in a different position or different channel or different chromatogram might not.

Deletion, insertion, and substitution errors

The search regions described above are computed using information about the length of the repetitive regions in the consensus sequence and individual reads. The consensus multiplicity (*cm*) of a consensus sequence element *c*, or the associated slice, is equal to 1 if *c* is not an element of a single base repeat region and it is equal to *len* if *c* is a part of a single base repeat region, where *len* is the length of this repeat fragment. Note that we ignore gaps when looking for a single base repeat region. For example, in Figure 1 the consensus base C at position 13 (in the middle of the subsequence ATCAA) has consensus multiplicity 1 and the C at position 23 (in the middle of ACCCA) has consensus multiplicity 3.

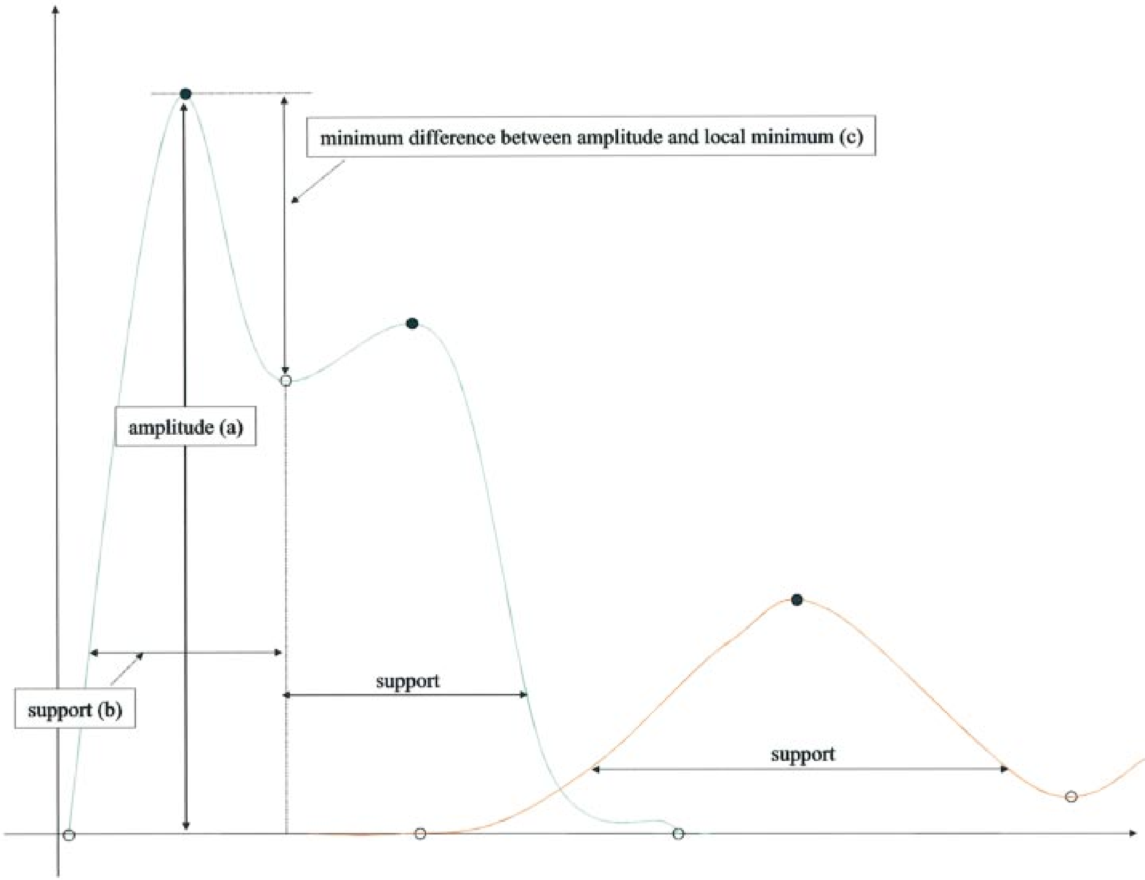

Figure 3. Signal parameters. Black dots on the signal curve correspond to local maxima and open circles correspond to local minima.

Read multiplicity is defined in essentially the same way as consensus multiplicity, but it is done on the level of a read, not the consensus sequence. More precisely, the read multiplicity (*rm*) of an element *c* of a read is equal to 1 if *c* is not a part of a single base repeat region within the read and *len* if *c* is a part of a single base repeat region, where *len* is the length of this repeat region. For example, in read [2] from Figure 1, the C in consensus sequence position 17 (in the middle of ACCTT) has read multiplicity 2, while the consensus sequence has a T at that position (in the middle of ACTTT) and its consensus multiplicity is 3. With these definitions, we can now describe the structure of the deletion, insertion and substitution modules of AutoEditor.

The deletion correction module of AutoEditor processes every slice containing a deletion element, i.e. the correction module processes every slice that contains a gap and whose homogeneous majority part consists of non-gap elements. (Note that the consensus base associated with such a slice is not a gap.) The correction module tests if $wrp[c] = cm$, where $wrp[c]$ is the number of well-resolved peaks in the consensus base channel within the search region and *cm* is the consensus multiplicity of the slice. If this condition is satisfied, the gap is changed by setting it equal to the consensus base. If $wrp[c] \neq cm$, $wrp[c] \neq 0$ and $cm > 1$, AutoEditor searches other reads of the selected slice and checks if the condition $wrp[c] = cm$ is satisfied in at least two of them (where two is a user-specified parameter). The reason AutoEditor searches the other reads to correct the deletion error is that the conditions $wrp[c] \neq cm$, $wrp[c] \neq 0$ and $cm > 1$ indicate that we are dealing with an unresolved peaks error. This in turn means that the chromatogram does have a signal in the search region, but the base caller likely made an error in reporting the correct number of bases. Since we assume that the consensus sequence is a reasonable estimate of the truth (the problems of misassembly are addressed below), we use the other reads in a slice to suggest how to correct base caller errors. For example, in the case of the deletion element in the slice TTC* at position 17 in the assembly fragment in Figure 1, the deletion correction module checks if there are three well-resolved ‘T’ peaks in the chromatogram region corresponding to the sequence CTTA in read [3]. If the number of well-resolved peaks is three (as in the case of a signal shift error, as depicted in Fig. 2), AutoEditor replaces CTTA with CTTTA. If the number of well-resolved peaks is non-zero but different from three (as it would be in the case of an unresolved peaks error, as depicted in Fig. 2), AutoEditor analyzes the two other reads to check if they contain three well-resolved ‘T’ peaks in the search region. If this is the case, AutoEditor again replaces CTTA with CTTTA.

In the insertion correction module, AutoEditor checks slices in which the minority of reads (usually just one) have a base inserted with respect to the majority. It tests whether $wrp[c] = cm$, where $wrp[c]$ and *cm* are computed in the consensus base channel on the read(s) containing the insertion base. If this

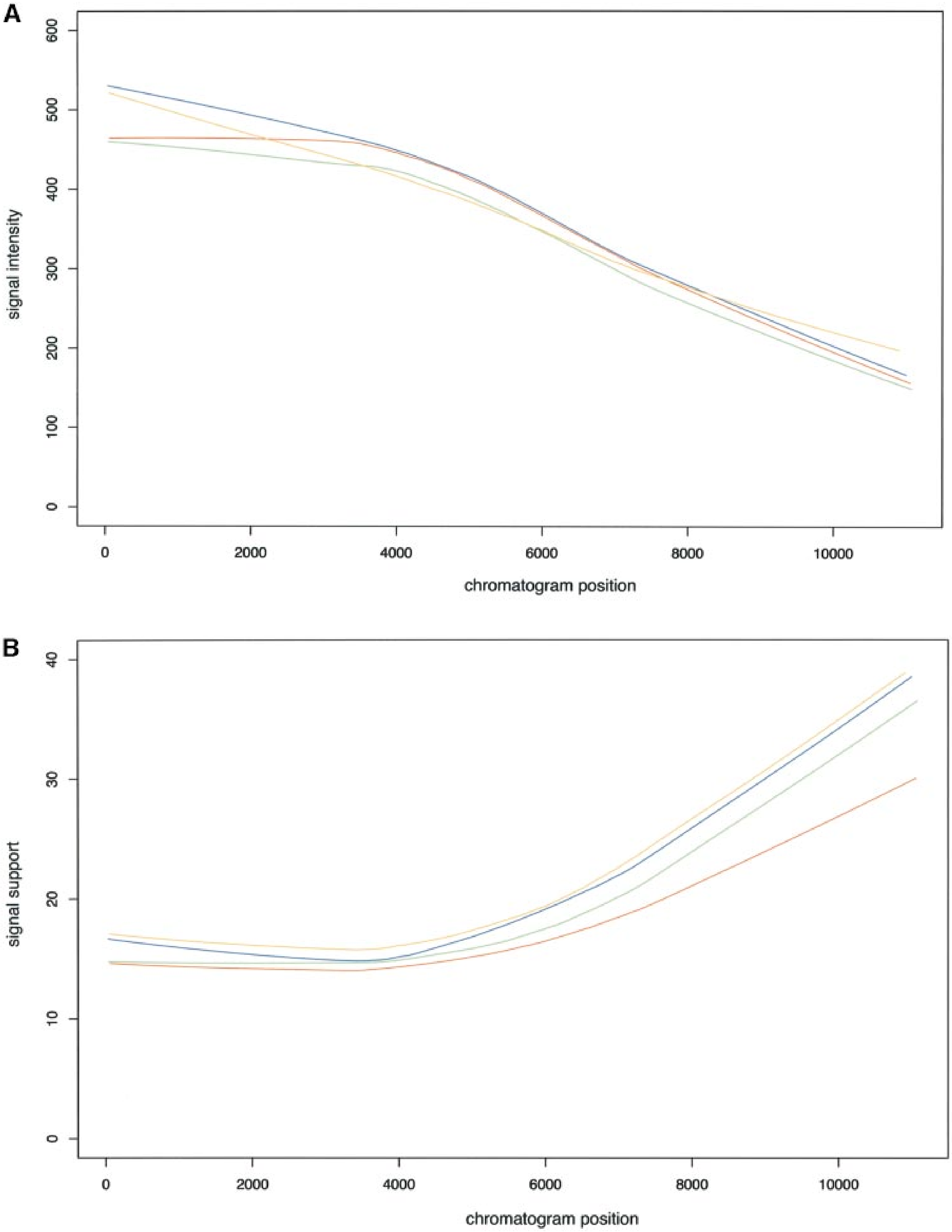

Figure 4. Dependency of signal intensity (**A**) and signal support (**B**) on the position of the signal within a chromatogram. Each line corresponds to a different channel with the color coding as follows: A, green; T, red; C, blue; G, orange.

condition is true, the insertion is considered to be valid and no change is made. If *wrp*[*c*] ≠ *cm*, the inserted base is changed to a gap. Searching other reads is done in the same fashion as in the deletion module. If all insertions in a slice are corrected, the slice is removed from the assembly, because the edited slice will then consist entirely of gaps. For example, in the case of the insertion base in the slice ***C at position 22 in Figure 1, the insertion correction module checks if there are four well-resolved 'C' peaks in the chromatogram region corresponding to the sequence ACCCCA in read [3]. If four well-resolved peaks are found, AutoEditor leaves the insertion element intact. If three or fewer well-resolved peaks are detected, AutoEditor takes this as an indication of an unresolved peaks error and uses the other reads to determine if the insertion is a base caller error.

In the substitution correction module, AutoEditor first checks if $wrp[s] = rm$, where $wrp[s]$ is the number of well-resolved peaks in the substitution base channel and rm is the read multiplicity of the substitution base. If this condition is true, AutoEditor considers the substitution to be valid and does not make any corrections. Otherwise, AutoEditor checks if $wrp[c] = cm$, where $wrp[c]$ is the number of well-resolved peaks in the consensus base channel. If this latter condition is true, then AutoEditor edits the substitution base to make it equal to the consensus base. If $wrp[c] \neq cm$ and $wrp[c] \neq 0$, AutoEditor uses other reads to confirm that the substitution element should be changed to the consensus base.

The use of other reads in the process of correcting base caller errors is easy to justify for clonal, haploid genomes, where the expected rate of polymorphism is close to zero. Even in these projects, though, one may encounter situations where the number of discrepant bases in a slice is nearly as large as the majority, especially in the case of misassembly. In this case the consensus base is less likely to represent the true underlying DNA and, therefore, we wish to avoid using other reads when the number of discrepant elements is too large. To capture this intuition, AutoEditor will not use other reads to correct unresolved peaks errors unless the number of discrepant elements within the slice is below a certain threshold (default 5) and the fraction of discrepant elements compared to the number of elements in the homogeneous majority part is less than a user-specified threshold (default 0.34).

RESULTS

To demonstrate AutoEditor's performance, we ran it on 26 genome projects, all done at TIGR in the period 1999–2003. These include 24 bacterial species and one eukaryote (the malaria-causing parasite *Plasmodium vivax*). For each genome we conducted the following experiment. First, we collected all the sequences generated by each WGS project and assembled them using the Celera Assembler (11). The assembler creates a set of contiguous DNA sequences (contigs) and a multiple alignment showing how each sequence maps to its contig. The average depth of coverage for all of our test genomes was 8.5, meaning that each position was covered on average by 8.5 individual sequences. We scanned all columns of this multiple alignment and identified columns (or 'slices') in which one or more of the sequences disagreed with the consensus for that column.

For each column containing a discrepancy, AutoEditor attempts to correct the discrepancy by changing it (if there is evidence for it) to make it match the consensus base. These discrepancies represent one of two possibilities: a base calling error or a true polymorphism in the input data. AutoEditor attempts to correct every discrepancy and those that go uncorrected stand as potential polymorphisms. Table 1 shows how many corrections AutoEditor makes for the 26 genomes in our study.

The projects used in this study contain over 109 million base pairs (Mb) of consensus sequences and the total length of all sequences is 927 Mb. Just over 3.4 Mb were contained in single coverage regions. Therefore, AutoEditor could not (by design) detect a disagreement between reads in these regions. About 4.4 Mb out of 927 Mb disagreed with the consensus assembly (column 2 in Table 1). If all of these were sequencing errors, the sequencing error rate would be 0.44%, which is less than half of the 1% error rate commonly reported for large-scale sequencing centers.

As described in Materials and Methods, AutoEditor re-analyzed the chromatogram (the underlying raw signal) for each of the 4 367 697 inconsistent bases and was able to re-call the base and correct the inconsistency for 3 470 821 bases, 79.5% of the total. Another way to count the corrections is to look at contig positions: these are columns (slices) in the multiple alignments that comprise all the contigs. Because multiple disagreements may occur in one column, the number of contig discrepancies, 3 854 419, is smaller than the number of inconsistent bases. Of these positions, 3 198 082, or 83%, were corrected by AutoEditor. This leaves 656 337 positions with an inconsistency that AutoEditor could not correct, which represent either true polymorphisms or additional base calling errors. A search for polymorphisms would start with those bases from this final set that have the highest probability (according to the original base caller) of being correctly called.

The one outstanding question that needs an answer is: are the 'corrections' that AutoEditor makes truly correct? In order to answer this question, we need a standard of truth for genome sequences that has a very small error rate. Fortunately, such a standard is available, in the form of completely sequenced bacterial genomes. For all the genomes that TIGR finishes, every position of the genome is guaranteed to have at least two-fold coverage and every repeat region is carefully 'walked' to ensure the correctness of each repeat copy. Any positions with low coverage or low quality base calls are checked manually. The final genome cannot be guaranteed to be error free, but recent reports indicate that these genomes have no more than 1 error per 100 000 bases (7,8). Because AutoEditor made corrections in about 3% of all positions, the error rate of finished genomes is so much lower that these can be used as a reliable standard of truth.

For each genome in Table 1 that has been completely finished, we compared the AutoEditor corrections to the final sequence and counted any disagreements as potential mistakes by AutoEditor. Because the number of disagreements was so small, we checked many of them by hand. Most occurred in regions of misassembly, where the assembler constructed a non-optimal multiple alignment. The rest are borderline-type errors, when the three parameters of a signal (height, depth of local minima and support) are near their threshold levels. In Table 2 we show the results of this comparison.

As long as AutoEditor makes fewer than 50% errors, the total number of errors in a collection of sequences will be reduced. Fortunately, as shown in Table 2, the error rate of AutoEditor is far smaller than this, only 149 errors from over 1.3 million corrections or, equivalently, 1 error per 8828

Table 1. AutoEditor results on a collection of recent genome sequencing projects

Organism	Discrepancies	Corrected	%	Contig discrepancies	Corrected	%
Acidobacterium capsulatum	103 539	93 729	90.5	99 555	89 977	90.4
Neorickettsia sennetsu Miyayama	41 408	37 425	90.4	38 355	34 579	90.2
Bacillus anthracis Kruger B	317 745	284 503	89.5	296 222	264 646	89.3
Coxiella burnetii	131 183	117 232	89.4	118 723	105 562	88.9
Dichelobacter nodosus	83 804	73 547	87.8	76 766	67 900	88.5
Clostridium perfringens	71 928	62 822	87.3	66 546	59 929	90.1
Mycoplasma capricolum	17 805	15 444	86.7	16 574	14 584	88.0
Brucella suis	129 870	112 359	86.5	120 799	105 250	87.1
Plasmodium vivax	783 495	655 642	83.7	734 298	618 268	84.2
Pseudomonas fluorescens	234 264	194 771	83.1	224 049	186 276	83.1
Campylobacter jejuni	96 231	79 237	82.3	88 800	73 940	83.3
Fibrobacter succinogenes	243 270	196 150	80.6	208 790	175 294	84.0
Erwinia chrysanthemi	219 370	176 354	80.4	205 161	165 070	80.5
Mycobacterium smegmatis	433 105	346 503	80.0	363 017	309 774	85.3
Prevotella intermedia	118 857	94 162	79.2	110 750	87 931	79.4
Pseudomonas syringae	227 887	177 897	78.1	200 223	164 561	82.2
Silicibacter pomeroyi	156 130	116 907	74.9	148 006	112 093	75.7
Chlamydophila caviae	50 137	36 972	73.7	47 875	35 103	73.3
Wolbachia sp.	70 782	51 163	72.3	57 357	45 401	79.2
Burkholderia mallei	139 359	99 711	71.6	130 158	94 540	72.6
Streptococcus agalactiae	152 330	105 878	69.5	109 821	92 153	83.9
Streptococcus pneumoniae	53 566	36 557	68.3	43 093	33 432	77.6
Myxococcus xanthus	33 525	21 789	65.0	33 254	21 699	65.3
Dehalococcoides ethenogenes	71 587	46 416	64.8	61 878	42 649	68.9
Listeria monocytogenes	229 172	145 274	63.4	148 177	123 268	83.2
Streptococcus mitis	157 348	92 377	58.7	106 172	74 203	69.9
Total	4 367 697	3 470 821	79.5	3 854 419	3 198 082	83.0

The number of discrepancies is the total number of bases in individual sequences that disagree with the consensus. The number of contig discrepancies is the number of positions in all contigs in which at least one sequence disagrees with the consensus. This number is smaller because there are many cases where multiple sequences disagree at a single position. The *Wolbachia* sp. is an endosymbiont of *Drosophila melanogaster*.

Table 2. Comparison of AutoEditor corrections on 14 genomes to the finished sequence of those genomes

Organism	Read length	Corrections	AE errors
Listeria monocytogenes	37 420 828	145 274	4
Wolbachia sp.	11 446 011	51 163	0
Burkholderia mallei	47 407 080	99 711	28
Brucella suis	26 629 877	112 359	2
Streptococcus agalactiae	23 485 615	105 878	3
Coxiella burnetii	29 135 115	117 232	30
Campylobacter jejuni	15 013 845	792 37	11
Chlamydophila caviae	10 286 694	36 972	6
Dehalococcoides ethenogenes	10 724 521	46 416	12
Neorickettsia sennetsu Miyayama	8 805 232	37 425	0
Fibrobacter succinogenes	46 463 268	196 150	4
Mycoplasma capricolum	9 353 819	15 444	0
Prevotella intermedia	20 084 365	94 162	3
Pseudomonas syringae	50 369 232	177 897	46
Total	346 625 502	1 315 320	149

corrections. Thus it accomplishes a dramatic reduction in the total number of base calling errors for all of the genomes studied here. As we can see from Table 2, AutoEditor makes on average fewer than 4.3 errors per 10 Mb of sequence reads.

AutoEditor recomputes the consensus sequence of the input assembly. Therefore, contigs processed with AutoEditor will in some cases have their consensus sequences changed. We ran the following experiment to estimate how many of these changes are due to the AutoEditor error correction algorithm. We compared the consensus sequences re-called by AutoEditor before and after error correction using the 5 Mb genome of *Pseudomonas syringae*. We first recomputed the consensus using AutoEditor's consensus base caller. Then we ran the error correction modules and recomputed the consensus again. In total, 173 consensus positions changed, all of the form ⟨ambiguity code⟩ → ⟨non-ambiguous base or gap⟩.

Genome finishing

As described in the Introduction, a time consuming and costly part of genome finishing is the process of editing sequences to improve the overall assembly quality. Large-scale sequencing centers employ specialists known as 'finishers' or 'closure teams' who examine, by hand, many of the assembly positions where one or more sequences disagree. Because finishers do not necessarily examine every discrepancy, we also calculated, for the same set of genomes shown in Table 1, how many positions would require manual finishing using our own center's standards. Not every position would require manual editing; for example, if a consensus sequence is supported by eight high quality A residues with just one low quality G, the finishers need not review it. Of the 3.85 million positions in Table 1 containing at least one sequence discrepancy, 1 174 392 positions would require manual review by our standards. Of these, 85% were automatically corrected by AutoEditor. This illustrates how AutoEditor can not only improve the overall quality of sequences, but can also significantly lower costs and speed efforts towards finishing genomes.

DISCUSSION

AutoEditor is a second generation base caller that improves on the abilities of existing base callers such as Phred (1,2) and TraceTuner (http://www.paracel.com/products/tracetuner.php). The key to the success of AutoEditor is the use of a genome assembly, and the redundant sequences at most positions, to drive the correction algorithm. By using the multiple sequences covering each position to check one another, the algorithm not only identifies possible sequencing errors, but also knows what the 'truth' is likely to be, giving it a significant advantage over a traditional base caller.

AutoEditor not only improves the quality of the contigs within the assembly, it can also improve the assembly itself: typically, an assembly will contain many instances of contigs that overlap very slightly, but not enough for an assembler to detect. There are at least two reasons why this happens: first, the contigs may overlap, but the overlapping portions contain base calling errors. These errors reduce the percent identity in the overlap sufficiently that the assembler does not merge the contigs. Second, the contigs may overlap by only a few bases, too little for the assembler to put together. By correcting base calling errors, AutoEditor sometimes fixes the first problem and allows an assembler to make larger contigs. We are addressing the second problem by using AutoEditor to extend the 'good' portion of the sequencing read, which will create longer overlapping regions and improve assemblies further.

One of the most active areas of research today, in species ranging from bacteria to mammals, is the discovery and understanding of polymorphisms between closely related organisms. Polymorphisms range from large-scale rearrangements and large DNA insertions and deletions to SNPs [see for example (10)]. As interest in SNP discovery has grown, more scientists have looked at the details underlying the consensus sequences in the public archives; in particular, the scientific community needs to know the accuracy of every base in a genome in order to determine whether or not a SNP is genuine. For large genomes even a very small sequencing error rate will produce a huge number of false SNPs, swamping any attempt to study SNPs in those sequences. Even for small genomes, the sequencing error rate can overwhelm the true SNP rate if the two organisms are highly similar. AutoEditor promises to dramatically improve our ability to discover true SNPs, because it corrects over 80% of the incorrect base calls in an assembly.

We have already used AutoEditor for SNP discovery, most notably in our work on the anthrax bacterium, *Bacillus anthracis*. In the winter of 2001–2002, we sequenced two near-identical strains of anthrax in an effort to find any sequence differences that could be used as forensic markers (8). The reference genome was the Ames strain, originally isolated in 1981 in Texas. The second genome was an isolate taken from a patient, the first victim to die from the 2001 anthrax attacks. Our study discovered four single nucleotide differences between the two chromosomes, which are 5.3 Mb in length. As we pointed out in that study, a sequencing error rate of even 10^{-4} in each genome would produce more than 1000 false SNPs when the two genomes were aligned. To solve this problem, we used the deep coverage in the assemblies to reduce this error rate to essentially zero. The success of this strategy led to the development of AutoEditor, which is now routinely used in our SNP discovery process for bacterial genomes.

Code availability

AutoEditor is an open source system. The complete source code and documentation is freely available at http://www.tigr.org/software/autoeditor.

ACKNOWLEDGEMENTS

The authors want to thank Martin Shumway and Jessica Vamathevan for their help and support in the process of development of AutoEditor. This work was supported in part by the NIH under NIAID contract N01-AI-15447 and NLM grant R01-LM06845.

REFERENCES

1. Ewing,B., Hillier,L., Wendl,M.C. and Green,P. (1998) Base-calling of automated sequencer traces using phred. I. Accuracy assessment. *Genome Res.*, **8**, 175–185.
2. Ewing,B. and Green,P. (1998) Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res.*, **8**, 186–194.
3. Tammi,M.T., Arner,E., Kindlund,E. and Andersson,B. (2003) Correcting errors in shotgun sequences. *Nucleic Acids Res.*, **31**, 4663–4672.
4. Kececioglu,J. and Yu,J. (2001) Separating repeats in DNA sequence assembly. *Proceedings of the Fifth Annual International Conference on Computational Biology (RECOMB)*, Montreal, Canada, pp. 176–183.
5. Pevzner,P.A., Tang,H. and Waterman,M.S. (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl Acad. Sci. USA*, **98**, 9748–9753.
6. Batzoglou,S., Jaffe,D.B., Stanley,K., Butler,J., Gnerre,S., Mauceli,E., Berger,B., Mesirov,J.P. and Lander,E.S. (2002) ARACHNE: a whole-genome shotgun assembler. *Genome Res.*, **12**, 177–189.
7. Fleischmann,R.D. (2001) Single nucleotide polymorphisms in *Mycobacterium tuberculosis* structural genes—response to Dr. Musser. *Emerg. Infect. Dis.*, **7**, 487–488.
8. Read,T.D., Salzberg,S.L., Pop,M., Shumway,M., Umayam,L., Jiang,L., Holtzapple,E., Busch,J.D., Smith,K.L., Schupp,J.M. *et al.* (2002) Comparative genome sequencing for discovery of novel polymorphisms in *Bacillus anthracis*. *Science*, **296**, 2028–2033.
9. Patil,N., Berno,A.J., Hinds,D.A., Barrett,W.A., Doshi,J.M., Hacker,C.R., Kautzer,C.R., Lee,D.H., Marjoribanks,C., McDonough,D.P. *et al.* (2001) Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, **294**, 1719–1723.
10. Kruglyak,L. and Nickerson,D.A. (2001) Variation is the spice of life. *Nature Genet.*, **27**, 234–236.
11. Myers,E.W., Sutton,G.G., Delcher,A.L., Dew,I.M., Fasulo,D.P., Flanigan,M.J., Kravitz,S.A., Mobarry,C.M., Reinert,K.H., Remington,K.A. *et al.* (2000) A whole-genome assembly of *Drosophila*. *Science*, **287**, 2196–2204.