Algorithms for sequence analysis via mutagenesis

Jonathan M. Keith¹,∗, Peter Adams¹, Darryn Bryant¹, Duncan A. E. Cochran¹,², Gita H. Lala¹,² and Keith R. Mitchelson²,³

¹Department of Mathematics, ²Australian Genome Research Facility and ³Institute for Molecular Bioscience, University of Queensland, St. Lucia Qld. 4072, Australia

Received on January 18, 2004; revised on March 11, 2004; accepted on March 23, 2004
Advance Access publication May 14, 2004

ABSTRACT
Motivation: Despite many successes of conventional DNA sequencing methods, some DNAs remain difficult or impossible to sequence. Unsequenceable regions occur in the genomes of many biologically important organisms, including the human genome. Such regions range in length from tens to millions of bases, and may contain valuable information such as the sequences of important genes. The authors have recently developed a technique that renders a wide range of problematic DNAs amenable to sequencing. The technique is known as sequence analysis via mutagenesis (SAM). This paper presents a number of algorithms for analysing and interpreting data generated by this technique.

Results: The essential idea of SAM is to infer the target sequence using the sequences of mutants derived from the target. We describe three algorithms used in this process. The first algorithm predicts the number of mutants that will be required to infer the target sequence with a desired level of accuracy. The second algorithm infers the target sequence itself, using the mutant sequences. The third algorithm assigns quality values to each inferred base. The algorithms are illustrated using mutant sequences generated in the laboratory.

Availability: Software will be made available upon request.
Contact: j.keith1@mailbox.uq.edu.au

INTRODUCTION
Conventional Sanger sequencing has been a tremendously powerful technique, and is the basis of all past and current genome sequencing projects. However, despite the many technical advances that have made these projects possible, some DNAs remain difficult or impossible to sequence using currently available methods. Even in so-called ‘finished’ genomes, there are regions that are apparently unsequenceable. These problematic regions range in length from tens to millions of bases. Techniques for sequencing problematic DNAs are needed to bring these projects to true completion and to simplify and enable current and future sequencing projects.

Sequencing problems can arise for a variety of reasons, and at various stages of the sequencing process. For example, some DNAs are unclonable. A number of causes for this phenomenon have been identified (Razin et al., 2001), but it is still not fully understood. Some DNAs inhibit the proper functioning of sequencing enzymes or form secondary structures that interfere with the movement of the molecule through a gel. Other DNAs are prone to replication slippage during amplification by polymerase chain reaction (PCR), resulting in a blurred chromatogram trace.

Numerous methods that reduce the stability of duplex DNA have been used to overcome these problems. One such method is the introduction into the sequencing reactions of non-mutagenic, strand-destabilizing nucleotide analogues, such as dITP (Kawase et al., 1986; Bergstrom et al., 1997), dUTP, 7-deaza-dGTP (Fernandez-Rachubinski et al., 1990; Motz et al., 2000; Dierick et al., 1993) and N4-methyl-2′-deoxycytidine 5′-triphosphate (Li et al., 1993). Some of these are now included in the formulations of commercial sequencing kits. Some other methods are: the inclusion of denaturing chemicals (Varadaraj and Skinner, 1994), sulphones (Chakrabarti and Schutt, 2002) or DMSO (Seto et al., 1995); and shearing of DNA into smaller pieces to disrupt the motif. Improvement in sequencing can also be achieved in some refractory regions using alternative sequencing enzymes (Kukanskis et al., 2000) with modifications to cycling parameters, and by the use of dye terminators instead of dye-primers (Robbins et al., 1996). For a recent review of methods for handling problematic DNA, see Hunt et al. (2003).

The authors have recently developed a new technique with the potential to render a wide range of problematic DNAs amenable to sequencing (Keith et al., 2004b). The technique

∗To whom correspondence should be addressed.
is called sequence analysis via mutagenesis (SAM) because it involves the counter-intuitive initial step of generating a number of mutated copies of the problematic DNA prior to sequencing. The idea is that mutation alters the DNA in such a way that the characteristics causing sequencing difficulties are not present in (some of the) the mutants. The extent of modification required to achieve this will depend on the nature of the problem, but in some cases it will be large. The mutants are then sequenced by conventional methods, and the original sequence is inferred from the mutant sequences. Such an inference is possible provided that there is a random element in the mutation process, so that the original sequence information is not lost but merely distributed among several fragments. The authors have successfully implemented SAM in the laboratory and used it to sequence previously unsequenceable DNAs. A more detailed description, including laboratory aspects, is given in Keith et al. (2004b). One advantage that SAM has over most other methods of handling problematic DNA is that it attempts to correct the cause of the problem, rather than merely dealing with its effects. Another advantage is its wide range of applicability: it can in theory be used to correct any sequencing problem.

This paper focuses on data analysis aspects of the method. Three algorithms for data analysis are presented. The first algorithm uses a simple model to predict the number of mutants that will be required to achieve a desired level of accuracy. The second algorithm uses a more complex model to infer an original sequence given the sequences of the mutants. The third algorithm assigns quality values to each base in the inferred sequence. Use of the algorithms is demonstrated on simulated and real data. Computational problems raised by various extensions of the basic method and directions for future research are also discussed.

SYSTEM AND METHODS

All algorithms described in this paper were implemented in C. All simulations and computations were performed on PCs with 850 MHz Intel Pentium III processors and 256 MB RAM. Laboratory methods for generating and sequencing the mutants mentioned in this paper are discussed in Keith et al. (2004b).

PREDICTING THE NUMBER OF MUTANTS

In order to implement SAM efficiently, and to take best advantage of the fact that mutants can be sequenced in parallel, it is desirable to predict in advance the number of mutants needed to achieve the required level of accuracy in the inferred sequence. If too few mutants are generated and sequenced initially, the accuracy of the inferred sequence will be too low and time had to be spent on processing additional mutants. If too many mutants are sequenced initially, resources would have been wasted. In this section, we develop a simple model for making such predictions.

The model is based on the following assumptions. (Note that these assumptions are used only for predicting the number of mutants required, not for inferring the original sequence.) First, we assume that the mutants differ from the target only by substitutions; there are no insertions or deletions. Consequently, if the mutants are aligned without gaps, each column of the alignment corresponds to a unique base in the target. Second, we assume that the substitutions are independent random events, and that the probability $p(y|x)$ that base $x$ will be mutated to base $y$ is known for all $x$ and $y$ in the alphabet $\Sigma = \{A, C, G, T\}$. Third, we assume that these probabilities are the same for all mutants. Finally, we assume that the four bases are present in the target in approximately equal proportions. The model presented here can be generalized in a straightforward manner to avoid making the latter two assumptions. However, we make them here to simplify exposition of the method.

Because there are no insertions or deletions, the problem of inferring the original sequence can be separated into the sub-problems of inferring an original base for each column of the gap-free alignment. From a Bayesian perspective, the posterior probability that base $x$ was the original base corresponding to a given column of the alignment is:

$$p(x|y_1, \ldots, y_n) \propto p(x) \prod_{z=A,C,G,T} p(z|x)^{n_z},$$

where $p(x)$ is the prior probability that the original base was $x$, $n$ is the number of mutants, $y_i$ is the character in row $i$ of the column under consideration and $n_z$ is the numbers of times base $z$ appears in the column. Note that $n_A + n_C + n_G + n_T = n$. We take the prior probabilities to be $p(x) = 0.25$ for all $x$. Consequently, the prior probabilities will cancel out of our calculations and can be ignored. The most probable original base can now be determined by evaluating the above expression for each $x \in \Sigma$ and selecting the base which returns the largest value. If the largest value is shared by two or more bases, we say that the original base is indeterminate. The posterior probabilities need not be normalized because we are only interested in their relative values.

The most probable original base is not necessarily the correct original base, and what we want to determine is the probability $\rho$ that this procedure results in a base being misidentified or indeterminate. If the original base was $x$, then the probability that the numbers of each base observed in the corresponding column of the alignment are $n_A, n_C, n_G$ and $n_T$ is given by the multinomial expression:

$$M(n_A, n_C, n_G, n_T|x)$$

$$\equiv \frac{n!}{n_A!n_C!n_G!n_T!} \times \prod_{z=A,C,G,T} p(z|x)^{n_z}.$$

The probability $\rho_x$ that base $x$ will be misidentified or indeterminate is therefore the sum of $M(n_A, n_C, n_G, n_T|x)$ over
The observed proportions of each type of substitution. To aligning mutants to a known original sequence and counting at a medium concentration. The estimates were obtained by substitution probabilities induced by the nucleotide analogue dPTP infrequently) and base composition will not be uniform. To note that the vertical axis is scaled logarithmically.

**Fig. 1.** Error probabilities for medium concentration dPTP mutagenesis.

Values of \(n_A, n_C, n_G\) and \(n_T\) for which this occurs, i.e.

\[
\rho_x = \sum_{(n_A, n_C, n_G, n_T) \in M_x} M(n_A, n_C, n_G, n_T|x) \times \delta_x(n_A, n_C, n_G, n_T),
\]

where \(M_x = \{(n_A, n_C, n_G, n_T) : n_A + n_C + n_G + n_T = n\}\) and \(\delta_x(n_A, n_C, n_G, n_T)\) equals one if the original base is misidentified or indeterminate and zero otherwise. Assuming equal proportions of each base, the overall probability of error \(\rho\) is the average of \(\rho_A, \rho_C, \rho_G\) and \(\rho_T\).

The following example illustrates this method. Suppose that mutants are generated using the substitution probabilities shown in Table 1. The value in the row labelled \(x\) and column labelled \(y\) is \(p(y|x)\). These are rough estimates of the substitution probabilities induced by the nucleotide analogue dPTP at a medium concentration. The estimates were obtained by aligning mutants to a known original sequence and counting the occurrences of each type of substitution.

Figure 1 (diamond markers) shows the dependence of \(\rho\) on the number of mutations for these substitution probabilities. The curves shown in the figure are merely visual aids for connecting series; they do not imply that the data can be interpolated. Note that the vertical axis is scaled logarithmically.

In practice, insertions and deletions will occur (although infrequently) and base composition will not be uniform. To test whether the above calculations are sensitive to such violations of the assumptions, the following simulations were performed. Short (400 nt) fragments of sequence were selected at random from a large database of human genomic sequence. The overall composition of the database is \(\sim 43\%\) GC, although composition of individual fragments varies considerably. For each fragment, mutants were simulated using the mutation probabilities shown in Table 2. These are similar to the values in Table 1, but allow for small probabilities of insertions, deletions and transversions. Probabilities of deletion for each base are in the last column of the matrices; probabilities of insertion are in the last row. The number of mutants generated varied from 2 to 10.

The algorithm for inferring an original sequence (see below) was then applied to the mutant sequences. The inferred sequence was compared with the known original sequence, and the number of errors calculated. The number of errors was quantified by the edit distance between the two sequences. The results are shown in Figure 1 (square markers). Each data point in this graph represents an average over \(\sim 1000\) simulations. The observed proportions of errors for the simulated data are similar to the calculated values, despite small violations of the assumptions.

An important issue that we have not considered here is the extent of modification required to render a problematic DNA amenable to sequencing. The extent of modification can be controlled by varying laboratory parameters such as reactant concentrations. Obviously, one would like the mutants to differ as little as possible from the original sequence, so that the original sequence can be inferred using a small number of mutants. However, one generally does not know in advance what minimal mutation intensity is required to remove a particular sequencing problem. One way to determine this would be to generate a number of mutants using different mutation intensities and attempt to sequence them. However, it may be more efficient to generate high intensity mutants by default and accept that a larger number of mutants will be required. This matter requires further investigation.

### INFERRING THE ORIGINAL SEQUENCE

Once the mutants have been generated and sequenced, the next task is to infer the original sequence. We describe the

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.99</td>
<td>0.00</td>
<td>0.10</td>
<td>0.00</td>
</tr>
<tr>
<td>C</td>
<td>0.00</td>
<td>0.96</td>
<td>0.00</td>
<td>0.04</td>
</tr>
<tr>
<td>G</td>
<td>0.04</td>
<td>0.00</td>
<td>0.96</td>
<td>0.00</td>
</tr>
<tr>
<td>T</td>
<td>0.00</td>
<td>0.08</td>
<td>0.00</td>
<td>0.92</td>
</tr>
</tbody>
</table>

### Table 2. Modification of Table 1 to allow insertions, deletions and transversions

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.898</td>
<td>0.001</td>
<td>0.098</td>
<td>0.001</td>
</tr>
<tr>
<td>C</td>
<td>0.001</td>
<td>0.958</td>
<td>0.001</td>
<td>0.038</td>
</tr>
<tr>
<td>G</td>
<td>0.038</td>
<td>0.001</td>
<td>0.958</td>
<td>0.001</td>
</tr>
<tr>
<td>T</td>
<td>0.001</td>
<td>0.078</td>
<td>0.001</td>
<td>0.918</td>
</tr>
<tr>
<td>—</td>
<td>0.002</td>
<td>0.002</td>
<td>0.002</td>
<td>0.002</td>
</tr>
</tbody>
</table>

### Table 1. Medium concentration dPTP substitution probabilities

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.90</td>
<td>0.00</td>
<td>0.10</td>
</tr>
<tr>
<td>C</td>
<td>0.00</td>
<td>0.96</td>
<td>0.00</td>
</tr>
<tr>
<td>G</td>
<td>0.04</td>
<td>0.00</td>
<td>0.96</td>
</tr>
<tr>
<td>T</td>
<td>0.00</td>
<td>0.08</td>
<td>0.00</td>
</tr>
</tbody>
</table>
main features of our algorithm for drawing such inferences in this section. Aspects of the algorithm have been described in a more general context elsewhere (Keith et al., 2002, 2003a,b). Here, we discuss the use of the algorithm in the context of SAM and illustrate the method using mutant sequences generated in our laboratory.

If there are no insertions or deletions, it is reasonable to ask whether the original sequence could be reconstructed by simply taking the consensus base at each position. The problem with this is that it does not take into account the fact that certain mutations are more likely to occur than others. For example, suppose that there is no A, two Cs and one T observed at a particular position in six mutants. If A to T mutations are unlikely compared with C to T mutations, and moreover if A to C mutations are unlikely compared with C to A mutations, then a C may be the more probable original base at that position, despite the majority of As. The relative probabilities of different substitutions can be estimated experimentally for a specific mutagenesis protocol, and our algorithm takes these into account.

Insertions and deletions do occur in practice, so these also need to be taken into account. A natural way to do this is to first perform a multiple sequence alignment. However, because SAM is aimed at problematic sequences, which are often highly repetitive, there could be substantial uncertainties in such an alignment. We therefore decided to infer the original sequence directly, without first constructing an alignment, thus removing a potential source of bias from the inference.

The algorithm is based on a Bayesian probabilistic model of the process by which mutations are introduced into the sequence. The first step in building the model is to assign a prior probability to the hypotheses that the original sequence was \( X \), for each finite sequence \( X \). We take this to be:

\[
p(X) = \begin{cases} 
\frac{1}{(L_{\text{max}} + 1)4^L}, & \text{if } 0 \leq L(X) \leq L_{\text{max}} \\
0, & \text{otherwise},
\end{cases}
\]

where \( L(X) \) is the length of \( X \) and \( L_{\text{max}} \) is the maximum sequence length that we consider. A very conservative value for \( L_{\text{max}} \) is the sum of the mutant lengths; a less conservative value is the length of the longest mutant. This prior probability effectively assumes that sequences of the same length are equally probable, and that the prior distribution of sequence length is uniform on the set \( \{0, 1, \ldots, L_{\text{max}}\} \).

Next, we require a model for the generation of the mutant sequences \( Y_1, \ldots, Y_n \). The model assumes that each mutant is generated by applying the following algorithm to the original sequence \( X \). The characters of \( X \) are numbered in order from left to right and a termination character is appended at the right end.

1. Set \( i := 1 \).
2. Consider an insertion immediately to the left of character \( i \) of \( X \).
3. If an insertion was made at Step 2,
   (a) Set \( i := i + 1 \).
   (b) Go to Step 2.
4. If character \( i \) is not the termination character,
   (a) Consider deleting or substituting character \( i \) of \( X \).
   (b) If a deletion was not made at Step 4a, set \( i := i + 1 \).
   (c) Go to Step 2.

Two assumptions are implicit in this algorithm: first, that mutations are generated by introducing a number of independent random point mutations into the original sequence; and second, that these mutations are introduced in order from left to right. The parameters of the model are the probabilities of each of the 12 possible substitutions, the probabilities of deleting each base and the probabilities of inserting each base. There are thus 20 independent parameters. These depend on the mutagen used, and are determined experimentally. A different set of parameters may be used for each mutant if different mutagens were used to generate them.

This model can be regarded as a hidden Markov model (HMM) with \( 2L(X) + 3 \) states. (See Rabiner (1989) for a tutorial on HMMs.) There are \( L(X) \) deletion/substitution states, corresponding to the \( L(X) \) characters of \( X \), and \( L(X) + 1 \) insertion states, corresponding to positions between characters and at the ends of \( X \). There is also an initial state and a final state, neither of which produce output. Transition probabilities between states are fixed by the experimentally determined parameters of the model. Observation symbol probability distributions for each of the deletion/insertion states depend on \( X \) and the experimentally determined parameters. Observation symbol probability distributions for the insertion states are fixed by the experimentally determined parameters. Thus, the HMM is fully specified by identifying \( X \).

An alignment of the mutant sequence \( Y_k \) to a candidate original sequence \( X \) may be interpreted as a possible set of point mutations by which \( Y_k \) could have been derived from \( X \). A character in \( Y_k \) aligned to a character in \( X \) may be interpreted as a substitution if the characters are different or as no mutation if the characters are the same. A character in \( Y_k \) aligned to a space may be interpreted as an insertion and a character in \( X \) aligned to a space may be interpreted as a deletion. A probability may thus be associated with each alignment, specifically the probability of the implied mutations under the model. The probability \( p(Y_k|X) \) that an original sequence \( X \) will give rise to a mutant \( Y_k \) may then be calculated by summing these probabilities over all alignments. This sounds like a formidable computational task, but in fact it can be implemented efficiently using dynamic programming (Keith et al., 2003a; Thorne et al., 1991). The (unnormalized) posterior probability that the original sequence was \( X \) can now be calculated using:

\[
p(X|Y_1, \ldots, Y_n) \propto p(X)p(Y_1|X) \cdots p(Y_n|X)
\]
in accordance with Bayes’ rule. Note that the constant of proportionality is not required as we are only interested in the relative probabilities of candidate original sequences. We use a simulated annealing algorithm to search for a sequence \( X \) that maximizes the posterior probability under this model. The search algorithm employs an efficient method of updating calculations of the posterior probability for each new candidate original sequence. Each iteration of the algorithm requires time \( O[L_{\text{max}} \sum_{k=1}^{n} L(Y_k)] \), where \( L_{\text{max}} \) is here redefined as the length of the longest candidate original sequence considered in that iteration. The space requirements are \( O[L_{\text{init}} \sum_{k=1}^{n} L(Y_k)] \) where \( L_{\text{init}} \) is the length of the initial candidate sequence for that iteration. Full details of the model and search algorithm are provided in Keith et al. (2002, 2003a, 2004a).

Our algorithm has strong similarities to HMM methods of multiple sequence alignment [introduced by Krogh et al. (1994)]. One difference is that the original sequence \( X \) is a parameter of our model; this enables this sequence to be inferred without ever having to construct an alignment. Consequently, the inference is not biased by the choice of alignment. Another difference is in our calculation and search techniques, which use recursion relations similar to those of the Forward–Backward procedure (a dynamic programming algorithm commonly used to estimate the probability of an observation for a given HMM) in the context of the string sampler developed by Keith et al. (2004a).

The model is computationally convenient, but how well does it model the processes it is intended to represent? To answer this question, we must digress briefly to describe the generation of mutants. The method used to generate the mutants used as examples in this paper is known as PCR mutagenesis (Zaccolo et al., 1996). We stress that it is only one of many ways that mutants can be generated for SAM. The key step in the process is amplification of the target DNA by PCR in the presence of nucleotide analogues. PCR is a cyclic process; in each cycle, double-stranded DNAs are separated and each strand becomes a template for the synthesis of a new double-stranded DNA. This results in exponential growth in the number of copies of the original DNA. However, the presence of nucleotide analogues disrupts this process. Nucleotide analogues resemble nucleotides, and may be incorporated into a synthesized strand in place of a nucleotide during any given cycle. During the next cycle, when the newly synthesized strands become templates, the nucleotide that pairs with a nucleotide analogue may not be the nucleotide that the analogue had originally paired with. The result is a single-base substitution. Such substitutions accumulate as the reaction proceeds, and the final products may be substantially different from the original molecule. The extent of modification can be controlled by altering the concentration of reactants or number of cycles.

The model described above closely represents the introduction of mutations in a single cycle of the PCR. It does not take into account the fact that there are multiple cycles. A better model would therefore represent each cycle by such a process, and would involve integrating over all possible intermediate products. However, this may be computationally infeasible. The model used here should provide a good approximation unless there are very many insertions and deletions.

Extensive simulations were performed to test our algorithm. Each simulation involved selecting a fragment of length 400 nt from a database of human DNA and simulating mutant sequences. The probability of substituting any given character was 0.2, with each of the three possible substitutions being equally probable. The probability of deleting a character was 0.01. The probability of inserting a character was also 0.01, with each of the four possible insertions being equally probable. Where insertions were made, a second insertion was considered with the same probability, and this process repeated until an insertion failed. We then inferred the original sequence using three different methods: the algorithm described here, an earlier non-probabilistic version of this algorithm (Keith et al., 2002) and Clustal W (Thompson et al., 1994). In each case, the error was taken to be the edit distance between the inferred sequence and the known original. Our probabilistic algorithm was given the insertion, deletion and substitution probabilities 0.01, 0.01 and 0.2, respectively, with no substitution biases. Our non-probabilistic algorithm does not allow for such input. Clustal W was tried with various gap penalties ranging from 1 to 20 and no substitution biases. Optimal results (for Clustal W) were achieved with a gap penalty of about eight. The Results for the three methods are shown in Figure 2. Each data point in the figure represents an average over ~1000 simulations. The comparison between our probabilistic algorithm and Clustal W is fair, since both algorithms are supplied with a uniform substitution matrix.

Fig. 2. Average edit distance between inferred and original sequences versus number of mutant sequences for: Clustal W with a gap penalty of 8.0 (triangular markers); a non-probabilistic algorithm due to the authors (diamond markers); and the probabilistic algorithm described in the text (square markers).
Indeed, it is more than fair, since the optimal gap penalty for use in Clustal W was determined using knowledge of the original sequence. Both of our algorithms outperformed Clustal W. The improvement can in part be attributed to the fact that our algorithms assume a star-like configuration of mutant relationships, whereas Clustal W assumes a binary tree. Additionally, our algorithms are not subject to biases introduced by deriving a consensus from a single (multiple sequence) alignment.

The following example illustrates the method. A short fragment of DNA (~200 nt) was obtained from chromosome 2 of the slime mold Dictyostelium discoideum (Dictyostelium Genome Sequencing Consortium, 2002). This fragment is of interest because it could not be sequenced using conventional methods, due to its high AT content. A total of 10 mutants were generated using PCR mutagenesis in the presence of the nucleotide analogue dPTP (Zaccolo et al., 1996). This analogue favours A to G and T to C transitions, and consequently tends to reduce AT content. Four mutants were generated using a low concentration of dPTP (one-third the concentration of the other dNTPs in the PCR) and six mutants were generated at a medium concentration (two-thirds the concentration of the other dNTPs). The mutants were sequenced by conventional methods without difficulty. Our algorithm was then used to infer the original sequence. The reconstructed sequence is shown in Figure 3, aligned to the mutant sequences.

The assumed mutation probabilities for the two concentrations are shown in Tables 3 and 4. Note that the parameters shown in Figure 3, aligned to the mutant sequences. Both of our algorithms outperformed Clustal W, due to its high A T content. A total of 10 mutants were generated using PCR mutagenesis in the presence of the nucleotide analogue dPTP (Zaccolo et al., 1996). This analogue favours A to G and T to C transitions, and consequently tends to reduce AT content. Four mutants were generated using a low concentration of dPTP (one-third the concentration of the other dNTPs). The mutants were sequenced by conventional methods without difficulty. Our algorithm was then used to infer the original sequence. The reconstructed sequence is shown in Figure 3, aligned to the mutant sequences.

Table 1. Model parameters used to infer the original sequence—low concentration dPTP

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>—</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.950</td>
<td>0.001 0.001 0.005</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
</tr>
<tr>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td></td>
</tr>
<tr>
<td>0.020</td>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
</tr>
<tr>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Model parameters used to infer the original sequence—low concentration dPTP

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>—</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.950</td>
<td>0.001 0.001 0.005</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
</tr>
<tr>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td></td>
</tr>
<tr>
<td>0.020</td>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
</tr>
<tr>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td>0.001 0.001</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. Inferred sequence (labelled ‘Inf’) aligned to four low concentration dPTP mutants (‘L1’–’L4’) and six medium concentration dPTP mutants (‘M1’–’M6’).
In this section, we present a method of assigning quality values to bases inferred using SAM. Our strategy is to compare the inferred original sequence with its immediate neighbours in sequence space; i.e. the set of all sequences that differ from it by a single-character insertion, deletion or substitution. These sequences are grouped, with each group containing sequences that differ from the inferred sequence at the same position. There is thus a group for each character in the inferred sequence and for each position between characters and at the ends of the sequence. The sequences in each group are compared with the inferred sequence by calculating relative probabilities. An implicit assumption made here is that all uncertainties about the inferred sequence pertain to single characters. Moreover, by examining only one base at a time, we implicitly assume that these uncertainties are in some sense independent. These assumptions are not true in general, but they are computationally convenient and are expected to result in reasonable approximations.

We require the following notation. Let \( L \) denote the length of the inferred original sequence \( X \) and let \( Y_1, Y_2, \ldots, Y_n \) denote the \( n \) mutant sequences as before. To simplify description of the method, it is convenient to append a termination character at the right end of \( X \), and refer to it as character \( L + 1 \) of \( X \). Let \( I_m \) be the set containing \( X \) and the four sequences obtained by inserting a character from \( \Sigma \) immediately to the left of character \( m \) of \( X \), for \( m = 1, \ldots, L + 1 \). Let \( D_m \) be the set containing \( X \), the sequence obtained by deleting character \( m \) of \( X \) and the three sequences obtained by substituting a character from \( \Sigma \) in place of character \( m \) of \( X \), for \( m = 1, \ldots, L \). Now, define:

\[
s(Z) = \prod_{k=1}^{n} p(Z|Y_k)
\]

for any finite sequence \( Z \). Define:

\[
i_m(Z) = \frac{s(Z)}{\sum_{Y \in I_m} s(Y)}
\]

for each \( Z \in I_m \) and for \( m = 1, \ldots, L + 1 \). Define:

\[
d_m(Z) = \frac{s(Z)}{\sum_{Y \in D_m} s(Y)}
\]

for each \( Z \in D_m \) and for \( m = 1, \ldots, L \). The probability that character \( m \) of \( X \) is the correct original character at that position, conditional on the rest of the sequence being correct, is \( d_m(X) \). Similarly, the probability that a character has not been omitted immediately to the left of character \( m \) of \( X \), conditional on the rest of the sequence being correct, is \( i_m(X) \). In keeping with the quality values output by phred (Ewing and Green, 1998), we assign a quality value of \(-10 \log_{10}[1 - d_m(X)]\) to character \( m \) of \( X \) and a quality value of \(-10 \log_{10}[1 - i_m(X)]\) to the absence of a character immediately left of character \( m \) of \( X \). We round both types of quality value down to the nearest integer. Quality values larger than 99 are not permitted; they are replaced by the value 99.

The method is defined by the following algorithm.

1. Calculate \( s(X) \).
2. Set \( m = 1 \).
3. While \( m \leq L + 1 \).
   (a) Calculate \( s(Z) \) for each \( Z \in I_m \setminus \{X\} \).
   (b) Calculate and output \( i_m(Z) \) for each \( Z \in I_m \).
   (c) Calculate and output \(-10 \log_{10}(1 - i_m(X))\)
   (d) If \( m \leq L \),
      (i) Calculate \( s(Z) \) for each \( Z \in D_m \setminus \{X\} \).
      (ii) Calculate and output \( d_m(Z) \) for each \( Z \in D_m \).
      (iii) Calculate and output \(-10 \log_{10}(1 - d_m(X))\)
   (e) Set \( m = m + 1 \).

This algorithm involves almost the same calculations as a single iteration of the algorithm for inferring an original sequence described in the previous section. It has the same time and space requirements, i.e. \( O[L_{\text{max}} \sum_{k=1}^{n} L(Y_k)] \) and \( O[L_{\text{init}} \sum_{k=1}^{n} L(Y_k)] \), respectively. It adds very little to the total time required to process the data.

To test the accuracy of quality values estimated by this method, we performed computer simulations. The simulations were identical to those used to test the first of our three algorithms, but the method of comparing inferred sequences to known original sequences was different. Instead of computing the edit distance, we constructed an alignment of the two sequences having the minimum number of implied single-character insertions, deletions and substitutions. Each point mutation implied by the alignment was considered an error at that sequence position, except that consecutive spaces in the inferred sequence were treated as a single deletion. Sequence positions were classified according to their assigned quality values, and the actual proportion of errors in each class was determined. In calculating these proportions, the results from simulations involving different numbers of mutants were pooled. Each error proportion \( p \) was then transformed into an error score, \(-10 \log_{10}(1 - p)\), and plotted against the corresponding quality value. The results for mutants generated using the mutation probabilities in Table 2 are shown in Figure 4. There was insufficient data to evaluate the accuracy.

### Table 4. Model parameters used to infer the original sequence—medium concentration dPTP

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>—</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.900</td>
<td>0.001</td>
<td>0.100</td>
<td>0.001</td>
<td>0.001</td>
<td>0.015</td>
</tr>
<tr>
<td>C</td>
<td>0.001</td>
<td>0.960</td>
<td>0.001</td>
<td>0.040</td>
<td>0.001</td>
<td>0.015</td>
</tr>
<tr>
<td>G</td>
<td>0.040</td>
<td>0.001</td>
<td>0.960</td>
<td>0.001</td>
<td>0.001</td>
<td>0.015</td>
</tr>
<tr>
<td>T</td>
<td>0.001</td>
<td>0.080</td>
<td>0.001</td>
<td>0.920</td>
<td>0.001</td>
<td>0.015</td>
</tr>
<tr>
<td>—</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>0.995</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Algorithms for SAM
of quality values higher than 50, but we did observe that no errors occurred at any base assigned a quality value higher than 55. The assigned quality values appear to underestimate the observed quality slightly, but it should be remembered that we detected errors using alignments that minimized the number of implied errors, and thus we may have over-estimated the observed quality. In any case, the assigned quality values give a good indication of the true quality of the base call.

Quality values for the Dictyostelium sequence used in the previous section are shown in Figure 5. There are 180 columns, corresponding to the functions \( d_1, \ldots, d_{180} \). The value at position 90 is shown in both parts of the figure. The values of \( i_1, \ldots, i_{181} \) are not shown, as they all take the maximum value 99. Note that quality values are high everywhere, except in one poly(T) region. There is uncertainty about whether there are 17 or 18 Ts in this region. Additional sequences are required to resolve this uncertainty.

**DISCUSSION**

In this paper, we have presented new algorithms for predicting the number of mutants required to achieve a desired level of accuracy using SAM and for assigning quality values to inferred sequences. The algorithm for inferring an original sequence has been described elsewhere, but its application in the context of SAM is new. Also new are the description of our model in terms of HMMs and the results obtained using Clustal W with an optimized gap penalty. There are many potential improvements and variations of the SAM technique that remain to be explored. This section mentions a few of these, with emphasis on computational aspects. We discuss two kinds of future development: first, algorithms for processing various kinds of SAM data and second, theoretical studies and computer simulations to investigate and optimize various forms of SAM.

There are a number of ways in which the algorithms described here could be improved. First, it may be possible to develop mutation models that more accurately represent the mutations arising from PCR mutagenesis. The models used here have a number of weaknesses. As we have already mentioned, they do not take into account the cyclic nature of the PCR used in PCR mutagenesis. They also implicitly assume that the probabilities of point mutations occurring are unaffected by local sequence content. Although this latter assumption is accurate most of the time, there is evidence that replication slippage occurs during the PCR in regions where a single base is repeated. Second, it may be advantageous to develop a model which separates errors introduced during sequencing from mutations introduced during mutagenesis. The current model ignores sequencing errors, and thus in effect combines the two processes. One way to take sequencing errors into account is to obtain quality values for the mutant sequences prior to reconstruction. It should be possible to modify the algorithms to take such quality values into account.

An important assumption in all of our algorithms is that mutations are randomly distributed. In reality, this is only an approximation. In some cases, a very specific mutation or small set of mutations is required to render a problematic DNA sequenceable. In such cases, all sequenceable mutants will have mutations in the critical regions. This increases the chance of bases being called incorrectly. There are a number
of ways of dealing with this problem. One is to ensure a high proportion of mutations, so that any mutant is likely to have some mutations in the critical region. However, this will not work if the critical region is a single base. Another approach is to attempt to detect probable mutation hotspots by data analysis, noting regions with unusually low quality values or where mutations that are unusual for the mutagenesis technique have occurred. Elsewhere (Keith et al., 2004b), we have presented data in which a hotspot can be detected in this manner precisely. In future, we intend to automate this process.

The mutant sequences used in SAM need not all be of the same type. Different mutants may be generated using different mutants or different mutagenic processes. All of the algorithms described in this paper can be readily generalized for such a scenario. There may be advantages to using different types of mutants. In particular, it may be possible to improve the accuracy of the inferred original sequence, without increasing the number of mutants. To see why this is so, consider the following example. Suppose that there are two different types of mutant. In one type, A → G and T → C mutations predominate, and in the other G → A and C → T mutations predominate. Two groups of hypothetical mutants of a short original sequence are shown in Figure 6.

Note that in parts of the sequence where substitutions occur in the first group of mutants, the second group reproduces the original sequence reliably and vice versa. Consequently, it should be possible to infer the original sequence with greater accuracy than would be possible had either type of mutant been used on its own. Optimal combinations of different mutant types need to be investigated, via theoretical studies or simulated reconstructions.

Additional examples of the application of SAM to real data are described elsewhere (Keith et al., 2004b). These include one in which SAM was applied to a DNA with known sequence, which was then successfully reconstructed. However, more work of this kind needs to be done to generate an extensive dataset for algorithm design and testing. In particular, such a dataset is needed to assess the validity of quality values assigned using our algorithm.

REFERENCES


