4828–4836 Nucleic Acids Research, 1994, Vol. 22, No. 22

A method for fast database search for all *k*-nucleotide repeats

Gary Benson* and Michael S.Waterman[1]
Department of Mathematics and [1]Department of Mathematics and Department of Molecular Biology, University of Southern California, DRB 155, 1024 W. 36th Pl., Los Angeles, CA 90089-1113, USA

Received April 14, 1994; Revised June 24 and Accepted August 15, 1994

ABSTRACT

A significant portion of DNA consists of repeating patterns of various sizes, from very small (one, two and three nucleotides) to very large (over 300 nucleotides). Although the functions of these repeating regions are not well understood, they appear important for understanding the expression, regulation and evolution of DNA. For example, increases in the number of trinucleotide repeats have been associated with human genetic disease, including Fragile-X mental retardation and Huntington's disease. Repeats are also useful as a tool in mapping and identifying DNA; the number of copies of a particular pattern at a site is often variable among individuals (polymorphic) and is therefore helpful in locating genes via linkage studies and also in providing DNA fingerprints of individuals. The number of repeating regions is unknown as is the distribution of pattern sizes. It would be useful to search for such regions in the DNA database in order that they may be studied more fully. The DNA database currently consists of approximately 150 million basepairs and is growing exponentially. Therefore, any program to look for repeats must be efficient and fast. In this paper, we present some new techniques that are useful in recognizing repeating patterns and describe a new program for rapidly detecting repeat regions in the DNA database where the basic unit of the repeat has size up to 32 nucleotides. It is our hope that the examples in this paper will illustrate the unrealized diversity of repeats in DNA and that the program we have developed will be a useful tool for locating new and interesting repeats.

INTRODUCTION

Repeating patterns make up a significant fraction of genomic DNA. For example, it has been estimated that from 30 to 50% of the human genome consists of repeats of one form or another. The exact function of many of these repeating regions is unknown but some may function as catalytic, regulatory or evolutionary sites (5,11,12,22,26). For example, the centromeric region of DNA controls the movement of the chromosome during cell division. This region, termed a *satellite*, consists of many contiguous copies of a species specific pattern and may serve as a protein binding site (9).

In some cases, repeating patterns have been implicated in human disease. A repeating three nucleotide pattern on the human X chromosome is sometimes replicated incorrectly, causing the number of repeats to balloon from 50 to hundreds or thousands (29). Individuals with this defect suffer from fragile-X mental retardation. Several other diseases are also now known to have their basis in huge expansions of different trinucleotide repeats (8,17,23).

Besides their importance in DNA function and expression, repeating patterns are useful laboratory tools. The number of copies of a pattern at a particular site on a chromosome is often variable among individuals (*polymorphic*). Such polymorphic regions are helpful in localizing genes to specific regions of the chromosome (linkage) and also in determining the probability of a match between two samples of genetic material via DNA fingerprinting (6,31).

Given the importance of repeating patterns and the exponential growth in the size of the DNA database, it is important to develop efficient methods for detecting repeats. In this paper, we describe a new program that does rapid scans of the database to find repeating regions where the basic unit of the repeat has size up to 32 nucleotides. Our program looks for *tandem repeats*, that is, a repeating region in which copies of the basic repeating unit occur one after the other. Besides database scans, our program will be useful as a tool for rapid identification of repeating regions in new entries to the database, thus facilitating more complete annotation of the sequences.

Several theoretical algorithms for finding tandem repeats have previously been described. One algorithm, (16), searches for tandem repeats when the criteria for similarity is either k or fewer mismatches (Hamming distance) or k or fewer differences (unit cost edit distance). Two other algorithms (2,14) search for *non-overlapping* regions in a sequence that give the best alignment score. They can be easily modified to find strictly tandem repeats. These two algorithms measure similarity by weighted operations for symbol replacement, insertion and deletion (18). None of the three can be modified to incorporate the more general scheme of length dependent gap penalties (10).

Actual use of these algorithms is problematic for several reasons. First, to analyze a single sequence of length n takes on

*To whom correspondence should be addressed at: Department of Biomathematical Sciences, Box 1023, The Mount Sinai Medical Center, One Gustave L. Levy Place, New York, NY 10029-6574, USA

the order of $n^2\log^2 n$ time, thus making these algorithms inefficient for database searches. Second, the algorithms are complex and difficult to program and rely on other complex algorithms. Third, in these algorithms, 'tandem repeat' means exactly *two* copies of a pattern rather than an unspecified number of copies. Since each 'repeat' may be composed of many copies of a basic unit, finding this unit would require additional computation.

An alternative, more computationally practical approach is provided by (19). In this method, a sequence is encoded using a data compression scheme. A region in the sequence that uses less than the expected number of bits for encoding is a 'simple sequence' which is composed of a mixture of fragments that occur elsewhere in the sequence. This method, too, has several limitations. First, the program does not specifically look for tandem repeats, and tends to report regions that are a mixture of repeats and other fragments. Second, in testing the program, we have found that it does *not* find short, approximate repeats that occur tandemly as many as 27 times. We estimate that such a region would occur with extremely low probability (randomly about once in 10^{15} random sequences; see Methods section). Third, the program output is just a description of the data compression encoding. There is no attempt to determine a basic repeating unit nor any other elements that are responsible for the short encoding. This makes comparison of repeats across sequences and species difficult and also precludes observations about molecular evolution (see Discussion section). Finally, no estimation of the statistical significance of the region is made other than that of the compressed encoding length.

Another algorithm that is useful for computing an alignment between a pattern and a sequence has been described by (7,20). The algorithm by itself is insufficient for *finding* repeating regions because the basic pattern of the region must be known in advance. In our method, we use this algorithm. More details are provided below.

In contrast to these other algorithms, our method is straightforward to describe and program. We have produced a working copy which quickly finds tandem repeats in database files containing tens of thousands of sequences. The program looks specifically for tandem repeats and can find such regions even if they contain only a few copies of a pattern. Additionally, each repeating region is reported with an alignment against a likely pattern. Using these patterns, it is easy for a user to spot similar repeats occuring within a single sequence or amongst several sequences and to make observations about the evolutionary history of the region.

The remainder of the paper is organized as follows. In the Methods section, we outline the procedures used in our program and discuss statistical problems associated with recognizing when patterns repeat significantly more often than expected at random. In the Discussion section, we present sample running times of our program and some examples of previously unidentified repeats from sequences in the primate genomic database. We also describe how observation of indel and replacement patterns in the repeating regions can suggest the evolutionary history of a repeating region, which may be significant when studying polymorphic regions.

METHODS

We begin with the definitions of some terms. A *pattern* is any particular sequence of bases. A *tandem repeat* is the concatenation of two or more copies of a pattern. Typically the copies are not exact, but contain various deletions, insertions and substitutions. A *period* is a basic unit of a tandem repeat. It is any one of the cyclic rotations of the pattern from which the tandem repeat is constructed. Often, we will speak of the *size* of a pattern or period. If a pattern or period has size = 7, then it consists of 7 bases. For example, for the pattern *ACG* of size 3, we could have the tandem repeat shown below:

A C G A C C A C G A G A C G A

Note that not all copies of the pattern are exact, and that there is not a complete copy of the pattern at the right end. The pattern has three possible periods: *ACG*, *CGA* and *GAC*, any of which could be considered the basic repeating unit.

Program outline

The program we describe in this paper searches for tandem repeats in DNA sequences. For each repeat found, the output consists of an alignment with a putative pattern together with a similarity score for the alignment. Typically, the user supplies the following information:

1. A *DNA database file* to be searched.
2. A *period size* for the patterns.
3. A *pattern detection parameter* (explained below).
4. A set of *similarity parameters* for weighting insertions, deletions and substitutions.
5. A *threshold value* for recognizing significant similarity scores.

The program works as follows. For each sequence in the database file:

1. Scan the entire sequence looking for *suspicious patterns*. Every time a suspicious pattern is detected, do the following:

(a) compute a *similarity score* for the pattern versus the sequence *in the region where the pattern was found.*
(b) if the similarity score exceeds a *threshold*:
   i. compute an *alignment* of the pattern and the sequence.
   ii. determine a *consensus pattern* from the alignment and recompute an alignment with the consensus pattern.
   iii. report sequence identification information and the alignment.

In the remainder of this section, we describe the major tasks of the program in more detail.

Detecting suspicious patterns

For very small period size, the number of possible patterns is small. For size = 3, the number of distinct pattern classes is 24, where cyclic rotations of a pattern are not considered distinct. That is, the three periods *CAT*, *ATC* and *TAC* are all considered the same pattern class. Recall that there are $4^3 = 64$, 3 letter words, so this grouping of periods into the same pattern class produces a significant reduction. Given such a small set of patterns, it would not be too costly in terms of time to search the entire database for each pattern.

As expected, the number of distinct patterns grows very rapidly with increasing period size. For size = 8, the number of distinct pattern classes is 8,230 and for size = 15, the number is 71,582,716! Obviously, for such an enormous set of patterns, it would be impossible to search each sequence for each pattern in a reasonable amount of time. Further, it would be *largely*

useless to do so since most patterns could not occur in a single sequence. Therefore, we need some method of *selecting patterns that actually occur* within a sequence. Further, we may want to select a pattern that *does not actually occur*, but is none the less the 'best' for a repeating region. These are perhaps the most subtle requirements to meet. We use the following approach.

In a region of tandem repeats, one would expect and indeed finds that in spite of the changes due to insertions, deletions and substitutions, small contiguous regions remain unaltered. We look for a repetition of such regions as illustrated in the following example:

T C T T G C A C T T A C...

Suppose we are searching for patterns with size = 7. If a 3 base string of nucleotides is repeated at an interval of 7, that *suggests* that the pattern between the repeats could be the period for a tandem repeat. In the example above, we find *CTT* repeated at an interval of 7 and conclude that *CTTGCA* could be a tandem repeat period. Note we are not looking specifically for *CTT*, only for any 3 base string that repeats at an interval of 7. A pattern found by this method we denote a *suspicious pattern*. Finding such a pattern triggers the next step of the algorithm which computes a similarity score for the pattern and the sequence in the region where the pattern is found.

There is nothing special about the *pattern detection parameter* 3 in the discussion above. It can be assigned any of the values 0, 1, 2, 3, etc. A zero value means every pattern of size 7 that occurs in the sequence is suspicious. Using zero will slow down the program because every pattern moves on to the next (costly) stage. Larger values for the pattern detection parameter require the tandem repeat to be highly conserved in some (small) part of its extent. We have found that values on the order of 3 to 8 work well with respect to time and with very little degradation in the number of repeats that the program actually finds.

Selecting a pattern that does not actually occur requires first computing an alignment. We put off the explanation of this until the section on consensus patterns.

Similarity scores

Having selected a suspicious pattern, we now want to determine if the pattern actually is part of a tandem repeat with that pattern as period. Here we will compute a similarity score for the pattern versus the local region of the sequence. The similarity score is a numerical rating of the similarity between the sequence and tandem repetitions of our suspicious pattern.

Computing the similarity scores for local alignment of two sequences $A=a_1a_2...a_n$ and $B=b_1b_2...b_m$ has been extensively studied (4,13,20,21,25). Using a method denoted *dynamic programming*, the computation involves filling out the entries of a rectangular array $S(i, j)$, where the row indices (i) correspond to bases in sequence A and the column indices (j) correspond to bases in sequence B. The value in each cell of the array is computed via a recurrence formula:

$$S(i, j) = \max \begin{Bmatrix} S(i-1, j-1) + \mu(i, j) \\ S(i-1, j) + \delta \\ S(i, j-1) + \delta \\ 0 \end{Bmatrix}$$

$$S(i, 0) = 0, \quad S(0, j) = 0$$

using *similarity parameters* μ and δ where $\mu(i, j)$ is the value given to a match or mismatch of a_i with b_j and δ is the value given to an insertion or deletion of a base (18). The score chosen depends upon whether it is better to match (or mismatch) a_i with b_j, insert a_i, delete b_j, or abandon the previous best alignment altogether and start over. (Note that this computation does not allow length dependent gap penalties (10). Such penalties could be easily introduced, but their inclusion would make the calculations approximately three times slower. We have chosen not to include them in our program because the goal is to *rapidly* identify repeats, not to produce the best alignment.)

To compute the similarity score, we need two sequences. Sequence A is the database sequence. Sequence B is the concatenation of some number, k, of copies of the pattern. Recall that we do not know, *a priori*, how many tandem copies of the pattern occur in the sequence (or in fact if more than one copy occurs). In order not to miss a very long string of repeats, k should be large. But, the dynamic programming computation takes time proportional to the *area* of the array. If k is large, the computation will be very time consuming, even for the most frequent cases where the pattern is not part of a tandem repeat.

Fortunately, there is an elegant solution to these problems. In the method of *wraparound dynamic programming* developed by (7,20) the B sequence consists of only a *single copy* of the pattern. Nonetheless, the maximum similarity score (and the corresponding alignment) can be computed.

In wraparound dynamic programming, the similarity scores for each row are computed in *two* passes through the row. We use the normal recurrence (defined above) for the cells $S(i, j)$ in a row i, except for the first cell $S(i, 1)$. Suppose the period has size p. In the first pass we use:

$$S(i, 1) = 0$$

and in the second pass we use:

$$S(i, 1) = \max \begin{Bmatrix} S(i-1, p) + \mu(i, j) \\ S(i-1, 1) + \delta \\ S(i, p) + \delta \\ 0 \end{Bmatrix}$$

This second calculation constitutes the wraparound technique, since the value in the first cell $S(i, 1)$ is calculated in part from previously computed values in the pth cells in rows i and $i-1$.

Once the maximum similarity score is determined using the wraparound technique, we compare it against a threshold value. If the score exceeds the threshold, then we continue with the next step of the program which is determining and reporting the alignment.

Calculating an alignment

An *alignment* is a representation of two sequences which indicates which bases are matched, substituted, inserted and deleted. For example the following is an alignment between the trinucleotide repeat *CGG* and one small stretch of the fragile-X FMR-1 gene:

```
CGTG  CGG  CAG  C-G  CGG    Sequence
CG-G  CGG  CGG  CGG  CGG    Pattern
```

Here, * indicates a substitution and − indicates an insertion or deletion. Remaining bases are matched.

Every similarity score $S(i, j)$ corresponds to (one or more) alignments of the two sequences A and B. If we choose the maximum score obtained in the array S, then an optimal alignment for the two sequences can be determined by starting at the maximum and tracing back through the array to determine where each value came from. The traceback, like the computation of scores, wraps around. Using similarity parameters $\mu(i, j) = 2$ for a match, $\mu(i, j) = -1$ for a substitution and $\delta = -2$ for an indel, we get the scores $S(i, j)$ below and the alignment shown above. (The boxes below represent the trace back from the high score of 21.)

		C	G	G
	-	0	0	0
C		2	0	0
G		0	4	2
T		2	2	3
G		2	4	4
C		6	4	3
G		4	8	6
G		8	6	10
C		12	10	8
A		10	11	9
G		11	12	13
C		15	13	11
G		13	17	15
C		17	15	16
G		15	19	17
G		19	17	21

Consensus patterns

A suspicious pattern P may produce an alignment that scores above the threshold, but P may not be the best pattern to align with the sequence in that region. In fact we often find that an extensive region of repeats contains many different suspicious patterns, with a range of alignment scores some above and some below the threshold.

Each calculation of an alignment is expensive and we want to minimize the number of times we examine the same stretch of sequence. In order to do this, we determine a *consensus pattern* from the alignment of P and the sequence in the following way. For each position i in P, we choose the majority sequence base aligned with that position. For example, in the following sequence fragment, $P = ACGTT$ is a suspicious pattern which produces the alignment:

```
  * *       *                     *   *
ACGAA ACGGTA -CGTT ACGT- AGGTA A
ACGTT ACG-TT ACGTT ACGTT ACGTT A
```

Yet, P is not the best pattern to align with the sequence. Selecting a majority base for each position of P, we get a consensus pattern $P_c = ACGTA$ which produces the alignment:

```
  *             *           *
ACGAA ACGGTA -CGTT ACGT- AGGTA A
ACGTA ACG-TA ACGTA ACGTA ACGTA A
```

Using the scoring scheme from the end of the previous section, (match = +2, substitution = −1, indel = −2), the first alignment gives a score of 27 and the second gives a score of 33, which is the best that can be done here.

Notice that selecting the consensus pattern can be more powerful than merely selecting suspicious patterns because it may produce a pattern that, as in the example, *does not actually occur in the sequence*. The consensus pattern is helpful in other ways:

i) Suppose we are searching for patterns of size 12. It is often useful to exclude patterns that have size 1, 2, 3, 4 or 6. (Each of these smaller pattern sizes will show up in a search for patterns of size 12.) For example dinucleotide repeats such as $(CA)_n$ occur frequently. In order to exclude them, we can easily test if a suspicious pattern is itself composed of repeats. Thus, the pattern $P_1 = CACACACA$ is composed of repeats of the pattern CA. A single mutation in a tandem repeat sequence can produce a suspicious pattern such as $P_2 = CCCACACA$ that has no internal repeats. Nonetheless, the consensus pattern will be P_1 and can be rejected.

ii) Although not illustrated by the example above, the consensus for a pattern position might be a deletion and the consensus between two pattern positions might be an insertion. If we find a majority of deletions at a position or a majority of insertions between two positions, the size of the consensus pattern will be adjusted. This may result in a different size than the one of interest. This information can again be used to reject a pattern or to notify that a different size pattern exists in a region.

Implementation details

The initial scan of the sequence for suspicious patterns is done in a Boyer-Moore (3) style, i.e. matches are checked from right to left. For example, suppose the pattern detection parameter is 4, the pattern size is 10 and we are testing sequence positions 25−29 versus 35−39. (We are looking for a suspicious pattern that starts in position 25.) First, we test positions 29 and 39 for equality. If they match, then positions 28 and 38 are tested, etc. Suppose positions 28 and 38 do *not* match. Then we can immediately move the testing positions to 29−34 versus 39−44 (testing for a suspicious pattern at position 29) since no match of length 4 can span the mismatch at positions 28 and 38 (i.e. no suspicious pattern can start in any of positions 25, 26, 27 or 28).

Each pattern is represented by a number in base 4. For example, $TACGA$ is represented by 30120_4. This is natural for a four character alphabet and also permits using fast bit shift operations instead of slower multiplications and divisions. After a suspicious pattern is selected, the *minimum cyclic permutation* of that pattern is found and used as the period for the dynamic programming step. The minimum cyclic permutation of $TACGA$ is $ACGAT = 1203_4$. Having calculated the alignment, we store on a list 1) the period and 2) the region of the sequence used. Every time we find a suspicious pattern, we first check the list so that the same period will not be tested again in the same region. This prevents recomputing the wraparound dynamic programming (wdp) step more than once for a suspicious pattern that appears repeatedly in the aligned region. Note also that using

the minimum cyclic permutation allows any rotated occurrence of the same pattern to be recognized.

Since a suspicious pattern may first appear in the middle of a region against which it aligns well, we do wdp both backwards and forwards from the location of the occurrence. Initializing the alignment score at zero, we do wdp backwards until all scores dependent on the occurrence trail off to zero. Reinitializing the alignment score to the maximum found in the backwards computation, we do wdp forward from the occurrence, until, again, all scores dependent on the occurrence trail off to zero. If the maximum score found in this second calculation exceeds the threshold, then we redo the entire wdp starting from the end of the backwards step and use this final calculation for the true score and alignment.

For a pattern that aligns with a score above the threshold, we proceed to the calculation of the consensus pattern as described in the previous section. When completed, we store the consensus period and region of the sequence used on the same list mentioned above and then jump to the end of the aligned region to restart the search for the next suspicious pattern.

Statistical considerations

After calculating the consensus similarity score, we want to estimate its statistical significance. Recall that our consensus score represents the best scoring pattern of all patterns of a given size. Karlin, *et al.*, (15), present formulas that can be adapted to give Poisson approximations to the significance of the maximum number of *exact* repeats of a given pattern. These approximations are a function of the letter composition of the pattern. There are several serious drawbacks to the direct application of these results. First, the formulas are not valid when applied to similarity scores derived from inexact matching. Second, our scores are derived from the maximum over *all* patterns of a given size, rather than from a single pattern. There is no known approximation for such a distribution, *even in the case of exact matching*. Finally, for larger pattern sizes, almost all of the patterns have negative score, the exceptions being those patterns actually or approximately occuring in the sequence. Although this sparseness is consistent with a Poisson approximation, from a practical standpoint it makes approximation by simulation infeasible because of the vanishingly small chance of picking a pattern that occurs in a sequence.

The approach we take to these difficult theoretical problems is to provide a Poisson approximation for our consensus similarity score by numerically estimating two critical parameters identified below. In our model, given a pattern size and similarity parameters, we assume that the empirical distribution function of maximum scores less than some value t in a sequence of size n can be approximated by a function of t and n. Our approximation function is:

$$F(n, t) = e^{-\gamma n \xi^t}$$

Models like this are motivated by the distribution of the length of the longest head run in coin tossing where $P(head) = \xi$. Since there are about $(1-\xi)n$ tails in n coin tosses, there are that many ways to begin head runs. Each of these head runs is t or longer in length with probability ξ^t. There are therefore an expected number $\lambda = (1-\xi)n\xi^t$ head runs at least t in length. Poisson approximation holds when λ is moderate or small in size, so the probability of no head runs of length t is

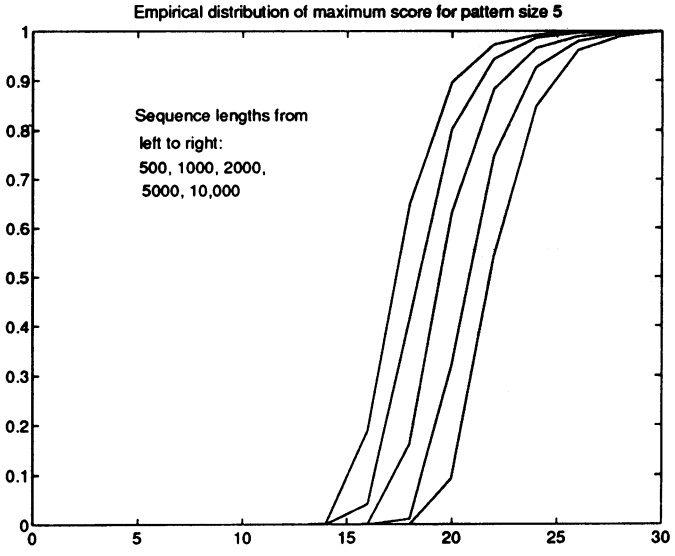

Figure 1. Simulated maximum similarity scores.

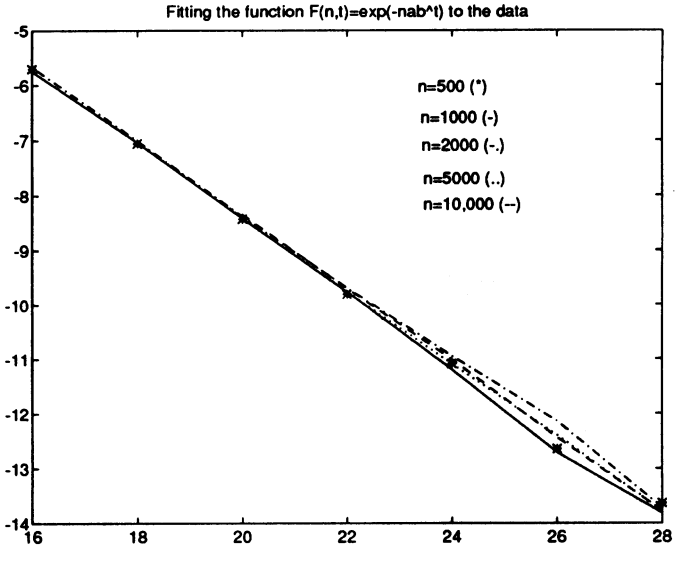

Figure 2. Transformation (log(−log(*data*)) of the data in Figure 1.

$$e^{-\lambda} = e^{-(1-\xi)n\xi^t}$$

In more general situations, the factor $(1-\xi)$ is replaced by another constant γ. For a general discussion see Arratia *et al.* (1), and for application to sequence matching see Waterman and Vingron (30).

We test the model by simulation. Each simulated sequence has *iid* letters for a given letter distribution. Initially, we have used equal probability for each letter. For each sequence, we run our repeats program with the pattern detection parameter set to zero and the consensus pattern computation disabled. This allows us to find the score for each pattern that occurs in the sequence.

Table 1.

Pattern size	Sequence length	Score
7	2000	27
7	5000	28
16	2000	45
16	5000	47
24	2000	61
24	5000	62

We collect the maximum score from each simulated sequence to calculate the empirical distribution.

In our tests, we used 5000 sequences each for sequence sizes n = 500, 1000, 2000, 5000, 10,000. The distribution data from a simulation for pattern size 5 and similarity parameters (μ = +2 for a match, μ = −6 for a substitution and δ = −9 for an indel) are shown in Figure 1. If the data fit our function *F(n, t)*, then after appropriate transformation, (log(−log(*data*)), the data are expected to fit a straight line. Figure 2 shows the transformed data. We see consistency of the data with the function *F(n, t)* over the entire range of sequence lengths tested.

We calculate the parameters γ and ξ from a linear regression. For example, the values for the parameter ξ calculated from the slopes of the regression lines for the data in figure 2 are .5045 for n = 500, .5010 for n = 1000, .5183 for n = 2000, .5100 for n = 5000 and .5067 for n = 10,000. The values for the parameter γ, calculated from the intercepts, are less stable. They range from a low of 91 to a high of 223.

Using the similarity parameters described above, and randomly generated sequences of size n = 500, we ran simulations for various pattern sizes. Scores as large or larger than those shown in Table 1 are likely to occur randomly about 5% of the time.

Given a score *t* for a repeating region in a sequence *S* of length *n*, we can estimate, from our simulations, the probability that a score at least that large will occur at random. The probability or p-value is computed by the function:

$$1 - F(n, t) = 1 - e^{-\gamma n \xi^t}$$

In the initial simulations we used *iid* letters, all with equal probability to compute *F(n, t)*. Clearly, sequence characters do not have this distribution. But neither does a region where we find repeats have the same distribution as the sequence as a whole. This suggests several ways to estimate the p-value. The difference is how we adjust the frequencies of letters in the random sequences in the simulation. We can use equal frequencies as above, we can use frequencies matching those of sequence *S* or we can use frequencies matching those in the repeating region within *S*. The first method will tend to give the smallest probabilities and the others higher probabilities. As an example, in the Discussion section we present a repeating region in the sequence HUMCAIIA01 in which the size 7 pattern is *ATCCCG*. The sequence has length approximately 3000. From simulation using random sequences with equal letter frequencies, we find that a score must be at least 28 to occur about 5% of the time. Using frequencies from HUMCAIIA01 (A = 0.214017, C = 0.293675, G = 0.278974, T = 0.213333) we similarly require a score of at least 28 for the 5% level. Using the much more skewed frequencies from the matched region (A = 0.1346, C = 0.596, G = 0.1346, T = 0.1346) we determine that a score must be at least 45 to occur about 5% of the time.

Table 2.

Pattern size	Detection parameter	Threshold	Time (min)
8	4	80	15
13	4	100	20
16	4	100	26
24	6	150	17
30	8	90	15

In the next section, we report sample repeats and their scores. We have estimated the probability of obtaining these or larger scores using the three methods described. In every case, the p-value is vanishingly small. (The exponent $-\gamma n \xi^t$ ranges variously from -10^{-8} for the case of frequencies matching those in the repeating region to -10^{-27} for the case of equal frequencies. These correspond to p-values from 10^{-8} to 10^{-27}.)

DISCUSSION

We judge the performance of the program by two characteristics. The first is the amount of time required to scan a database file and the second is its ability to find previously unnoticed repeats.

Time requirements

We tested the program on the GenBank primate sequences file (December 1993 release). This file contains approximately 29,000 sequences comprising some 28 million bases. Table 2 gives the times for runs of the program using various pattern sizes, pattern detection parameters and thresholds. For every run, the similarity parameters were μ = +2 for a match, μ = −6 for a substitution and δ = −9 for an indel. All the tests were run on a Sun Sparcstation 10.

Sample repeats found

Many of the features detected by our program are already annotated in the GenBank entries. They include centromeric regions, telomeric regions, repeat polymorphisms, microsatellites and minisatellites. Below we present some regions detected that are not included in the annotations. All the regions below come from the GenBank primate sequence database:

A size 8 pattern that occurs 45 times. It occurs in an intron of the Human int-2 proto-oncogene.

```
LOCUS       HSINT2      1
DEFINITION  Human int-2 proto-oncogene
Length: 11608

Alignment vs ACCCATCC

     (class: 05429)  Indices: 4504--4856  Score: 337

                                      *             *
4504 C  ACCCATCC  ACCCATCC  ACCCATTC  ACCCATCC  ACTCATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC

                              *         *
4544 C  ACCCATCC  ACCCATCC  ATCCATCC  ACTCATCC  ACCCATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC

                        *                 *         *
4584 C  ACCCATCC  ATCCATCC  A-CC-T--  ATCCATCC  ATCCATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC
```

```
                                    **    **
4620 C  ACCCATCC  ACCCATCC  TTCCATTT  ACCCATCC  A-CCATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC

                       *         *
4659 C  A-CC-T--  ATCCATCC  ATCCATCC  ACCCATCC  ACCCATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC

                 **                                  *
4695 C  ACCCATTT  ACCCATCC  ACCCATCC  ACCCATCC  ACCTATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC

            *         *          *                 *
4735 C  ATCCATCC  ATCCATCC  ACCTATCC  ACCCATCC  ACTCATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC

           * *     *   * *   *   *        **
4775 C  ACTCACCC  ATCCACCT  ATCCACCC  ACCCACTC  ACCCATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC

         *   *         **             *    *      *
4815 C  ATCCACCC  ACCCACTC  ACCCATCC  ATCCATAC  ACCTATC
   7 C  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATCC  ACCCATC


4855 C  A
   7 C  A
```

A size 16 pattern that repeats almost 6 times. It occurs between the cds for galactoside 3;(4)-L-fucosyltransferase and an ALU-like sequence.

```
LOCUS       HSAFUTF
DEFINITION  Human Lewis blood group locus mRNA for alpha(1,3/1,4)fucosyltransferase
Length: 2043

Alignment vs ACCTCGCCTGCTGGGG

     (class: 0392554410)  Indices: 1307--1395  Score: 130

           *                                 *
1307 GCCTGCTAGGG  ACCTCGCCTGCTGGGG  ACCTCGCCTGTTGGGG  A
   5 GCCTGCTGGGG  ACCTCGCCTGCTGGGG  ACCTCGCCTGCTGGGG  A

         *                 *                 * *
1351 CCTCACCTGCTGGGG  ACCTCACCTGCTGGGG  ACCTTGGCTGCTGG
   1 CCTCGCCTGCTGGGG  ACCTCGCCTGCTGGGG  ACCTCGCCTGCTGG
```

A size 7 pattern which repeats (exactly) over 7 times. It occurs in the human carbonic anhydrase II (CAII) gene.

```
LOCUS       HUMCAIIA01
DEFINITION  H.sapiens carbonic anhydrase II (CAII) gene, exons 1 and 2.
Length: 2925

Alignment vs ATCCCCG

     (class: 03414)  Indices: 1420--1471  Score: 104


1420 CCCCG  ATCCCCG  ATCCCCG  ATCCCCG  ATCCCCG  ATCCCCG
   2 CCCCG  ATCCCCG  ATCCCCG  ATCCCCG  ATCCCCG  ATCCCCG


1460 ATCCCCG  ATCCC
   0 ATCCCCG  ATCCC
```

A size 24 pattern that repeats almost 5 times. It occurs in the human GSTmu3 gene for a glutathione S-transferase Mu class protein.

```
LOCUS       HSGSTMU3
DEFINITION  Human GSTmu3 gene for a glutathione S-transferase Mu class protein,
Length: 1820

Alignment vs ACAGAGTGCTGATTGGTCCATTTT

     (class: 46542029704447)  Indices: 1407--1509  Score: 174

                        *
1407 GATTGGTCCATTTT  ACAGAGAGCTGATTGGTCCATTTT  ACAGAGTG
  10 GATTGGTCCATTTT  ACAGAGTGCTGATTGGTCCATTTT  ACAGAGTG

                *                          **
1453 CTGATTGGTCCGTTTT  ACAGAGTGCTGATTGGTCTGTTTT  ACAGAG
   8 CTGATTGGTCCATTTT  ACAGAGTGCTGATTGGTCCATTTT  ACAGAG


1499 TGCTGATTGGT
   6 TGCTGATTGGT
```

A size 5 pattern that repeats 19 times. It occurs in the human glutathione S-transferase (GST-pi) pi gene, 5′-flanking region.

```
LOCUS       HUMGSTPIA
DEFINITION  Human glutathione S-transferase (GST-pi) pi gene, 5'-flanking region
Length: 2669

Alignment vs AAAAT

     (class: 03)  Indices: 1845--1942  Score: 162

                                 * *
1845 AT  AAAAT  AAAAT  AAAAT  AACAC  AAAAT  AAAAT  AAAA
   3 AT  AAAAT  AAAAT  AAAAT  AAAAT  AAAAT  AAAAT  AAAA


1881 T  AAAAT  AAAAT  AAAAT  AAAAT  --AAT  AAAAT  AAAAT
   4 T  AAAAT  AAAAT  AAAAT  AAAAT  AAAAT  AAAAT  AAAAT


1915 AAAAT  AAAAT  AAAAT  AAAAT  AAAAT  AAA
   0 AAAAT  AAAAT  AAAAT  AAAAT  AAAAT  AAA
```

A size 13 pattern that repeats almost 16 times. It occurs in an intron of the human protein C inhibitor gene.

```
LOCUS       HUMPCI      1
DEFINITION  Human protein C inhibitor gene, complete cds.
Length: 15571

Alignment vs ACTCCACTCCTCC

     (class: 07675253)  Indices: 11738--11976  Score: 234

              *                    *   *
11738 CTCC  ATTCCACTCCTCC  ACTCCTCTCATCC  ACTCCACTCTACTC
    9 CTCC  ACTCCACTCCTCC  ACTCCACTCCTCC  ACT---C-C-ACTC

                   **
11782 CTCC  ACTCCACATCTCC  ACTCCACTCCTCC  ACTCCACTCCTCC
    9 CTCC  ACTCCACTCCTCC  ACTCCACTCCTCC  ACTCCACTCCTCC

              *
11825 ACTCCACTCATCC  ACTCCACTCCTCC  ACTCCACTCCTCC  ACTCC
    0 ACTCCACTCCTCC  ACTCCACTCCTCC  ACTCCACTCCTCC  ACTCC

                        *                 * *  **   *
11869 ACTCCTCC  ACTCCACTCCACTC  A-TCCACTCCACT  CTTCCATTC
    5 ACTCCTCC  ACTCCACTCCTC-C  ACTCCACTCCTCC  ACTCCACT-

              *                       *
11912 CACTCC  ATTCCACTCCTCC  ACTCCACTCTTCC  ACTCCA---CTC
    8 C-CTCC  ACTCCACTCCTCC  ACTCCACTCCTCC  ACTCCACTCCTC

           *
11953 C  ATTCCACTCCTCC  ACTCCACTCC
   12 C  ACTCCACTCCTCC  ACTCCACTCC
```

A size 32 pattern which repeats (exactly) about 4 times. It occurs in the human MYCL2 gene.

```
LOCUS       HUMMYCL2A
Length: 3854

Alignment vs ATATATATATGTATGTATATATGTATATATGT

     (class: 858995515859517755)  Indices: 2681--2814  Score: 268


    2681 ATATATGT  ATATATATATGTATGTATATATGTATATATGT  ATATAT
      24 ATATATGT  ATATATATATGTATGTATATATGTATATATGT  ATATAT


    2727 ATATGTATGTATATATGTATATATGT  ATATATATATGTATGTATATAT
       6 ATATGTATGTATATATGTATATATGT  ATATATATATGTATGTATATAT


    2775 GTATATATGT  ATATATATATGTATGTATATATGTATATAT
      22 GTATATATGT  ATATATATATGTATGTATATATGTATATAT
```

Evolutionary history

The ability to look at patterns of various sizes presents an interesting opportunity to reconstruct a likely evolutionary history for a repeating region. The mechanism for producing repeats is not yet understood, but may be due to unequal crossing over (24) or slippage during replication (27,28). Consider the following region in a retroviral DNA identified by our program as having a repeating unit of size 8:

```
LOCUS       AGMERLTR1
DEFINITION  African green monkey endogenous retroviral 5' LTR, segment 1 of 2.
Length:  612

Alignment vs AAACTTAG    Period size: 8

     (class: 0498)  Indices: 24--155  Score: 173

                      *  *                 *
     24 AAACTTAG  AAACTTAT  AGACTTAG  AAACTTAG  AGACTTAG
      0 AAACTTAG  AAACTTAG  AAACTTAG  AAACTTAG  AAACTTAG

                *                    *                 *
     64 AAACTTAG  AGACTTAG  AAACTTAG  AGACTTAG  AAACTTAT
      0 AAACTTAG  AAACTTAG  AAACTTAG  AAACTTAG  AAACTTAG

        *                    *        *  *
    104 AGACTTAG  AAACTTAG  AGACTTAG  AGACTCAG  AAACTTAG
      0 AAACTTAG  AAACTTAG  AAACTTAG  AAACTTAG  AAACTTAG


    144 AAAGCTTAG  AAA
      0 AAA-CTTAG  AAA
```

Careful observation reveals that *G* is periodically substituted for *A*. Such substitutions are unlikely to occur independently, suggesting that the 8bp unit *AAACTTAG* was first duplicated and then mutated to *AGACTTAG* and then the two copies were duplicated as a single 16 bp unit. Indeed, our program reveals just such a pattern (completely automatically) when the sequence is examined for a pattern of size 16, reducing the number of mismatches from 10 to 5:

```
Alignment vs AAACTTAGAGACTTAG    Period size: 16

     (class: 032645618)  Indices: 26--153  Score: 205

                      *
     26 ACTTAG  AAACTTATAGACTTAG  AAACTTAGAGACTTAG  AAACTT
     10 ACTTAG  AAACTTAGAGACTTAG  AAACTTAGAGACTTAG  AAACTT

                                                 *
     70 AGAGACTTAG  AAACTTAGAGACTTAG  AAACTTATAGACTTAG  AA
      6 AGAGACTTAG  AAACTTAGAGACTTAG  AAACTTAGAGACTTAG  AA

                        *  *  *
    114 ACTTAGAGACTTAG  AGACTCAGAAACTTAG  AAAGCTTAGA
      2 ACTTAGAGACTTAG  AAACTTAGAGACTTAG  AAA-CTTAGA
```

Further observation suggests that in one copy of this 16 bp unit, a *G* was mutated to a *T*, and then that copy was duplicated, accounting for an additional two mismatches.

CONCLUSION

We have described a new program for rapidly detecting repeating regions in DNA sequences where the period of the repeat has size up to 32 nucleotides. Our program can be used to quickly search the DNA database for a particular size period or to search a single sequence for all size periods. We combine our program with estimates of the statistics of sequence similarity scores in order to estimate the statistical significance of repeats that are detected.

It is our hope that the examples in this paper will illustrate the unrealized diversity of repeats in DNA and that the program we have developed will be a useful tool for locating new and interesting repeats.

The program written in C will be made available (email: gbenson@hto.usc.edu).

ACKNOWLEDGEMENTS

This work was partially supported by grants DMS-90-05833 and DMS-87-20208 from the National Science Foundation and by grants from the National Institutes of Health.

REFERENCES

1. Arratia,R., Goldstein,L. and Gordon,L. (1989) *Ann. Prob.*, **17**, 9–25.
2. Benson,G. (1994) In Crochemore,M., Gusfield,D. (eds) Combinatorial Pattern Matching Fifth Annual Symposium, Springer-Verlag Lecture Notes in Computer Science, **807**, 1–14.
3. Boyer,R.S. and Moore,J.S. (1977) *Comm. ACM*, **20**, 762–772.
4. Chang,W.I. and Lawler,W.L. (1990) *Proc.31st Annual IEEE Symposium on Foundations of Computer Science*, 116–124.
5. DeBustros,A., Nelkin,B.D., Silverman,A., Ehrlich,G., Poiesz,E. and Baylin,S.B. (1988) *Proc. Natl. Acad. Sci. USA*, **85**, 5693–5697.
6. Edwards,A., Hammond,H., Jin,L., Caskey,C. and Chakraborty,R. (1992) *Genomics*, **12**, 241–253.
7. Fischetti,V., Landau,G., Schmidt,J. and Sellers,P. (1992) *Third Annual Symposium on Combinatorial Pattern Matching*, 111–120.
8. Fu,Y.-H., Pizzuti,A., Fenwick,R.G.Jr., King,J., Rajnarayan,S.,Dunne,P.W., Dubel,J., Nasser,G.A., Ashizawa,T., DeJong,P.,Wieringa,B., Korneluk,R., Perryman,M.B., Epstein,H.F., Caskey,C.T. (1992) *Science*, **255**, 1256–1258.
9. Gall,J.G. and Atherton,D.D. (1974) *J. Mol. Biol.*, **85**, 633–634.
10. Gotoh,O. (1990) *Bull. Math. Biol.*, **52**, 359–373.

11. Hamada,H., Seidman,M., Howard,B.H. and Gorman,C.M. (1984) *Mol. Cell.Biol.*, **4**, 2622–2630.
12. Hellman,L., Steen,M.L., Sundvall,M. and Petterson,U. (1988) *Gene*, **68**, 93–100.
13. Hirschberg,D.S. (1975) *Comm. ACM*, **18**, 341–343.
14. Kannan,S. and Myers,G. (1993) In Apostolico,A., Crochemore,M., Galil,Z., Manber,U. (eds) Combinatorial Pattern Matching Fourth Annual Symposium, Springer-Verlag Lecture Notes in Computer Science, **644**, 74–86.
15. Karlin,S., Ost,F. and Blaisdell,B.E. (1989) In Waterman,M.S. (ed.) Mathematical Methods for DNA Sequences, CRC Press, 133–158.
16. Landau,G. and Schmidt,J. (1993) *Fourth Annual Symposium on Combinatorial Pattern Matching*, 120–133.
17. La Spada,A.R., Wilson,E.M., Lubahn,D.B., Harding,A.E. and Fischbeck,K.H.(1991) *Nature*, **352**, 77–79.
18. Levenshtein,V.I. (1966) *Soviet Phys. Dokl.*, **10**, 707–710.
19. Milosavljević,A. and Jurka,J. (1993) *CABIOS*, **9**, 407–411.
20. Myers,E.W. and Miller,W. (1988) *CABIOS*, **4**, 11–17.
21. Needleman,S.B. and Wunch,C.E. (1970) *J. Mol. Biol*, **48**, 443–453.
22. Pardue,M.L., Lowenhaupt, K., Rich,A. and Nordheim,A. (1987) *EMBO J.*, **6**, 1781–1789.
23. Richards,R.I. and Sutherland,G.R. (1992) *Nature Genet.*, **1**, 7–9.
24. Smith,G.P. (1976) *Science*, **191**, 528–535.
25. Smith,T.F. and Waterman,M.S. (1981) *J. Mol. Biol.*, **147**, 195–197.
26. Stallings,R.L., Ford,A.F, Nelson,D., Torney,D.C., Hildebrand,C.E. and Moyzis,R.K. (1991) *Genomics*, **10**, 807–815.
27. Strand,M., Prolla,T.A., Liskay,R.M. and Petes, T.D. (1993) *Nature*, **365**, 274–276.
28. Streisinger,G., Okada,Y., Emrich,J., Newton,J., Tsugita,A., Terzaghi,E. and Inouye,M. (1966) *Cold Spring Harb. Symp. Quant. Biol.*, **31**, 77–84.
29. Verkerk,A.J.M.H., Pieretti,M., Sutcliffe,J.S., Fu,Y.-H., Kuhl,D.P.A., Pizzuti,A., Reiner,O., Richards,S., Victoria,M.F., Zhang,F.,Eussen,B.E., van Ommen,G.-J.B., Blonden,A.J., Riggins,G.J., Chastain,J.L., Kunst,C.B., Galjaard,H., Caskey,C.T., Nelson,D.L., Oostra,B.A., Warren,S.T. (1991) *Cell*, **65**, 905–914.
30. Waterman,M. and Vingron,M. *Proc. Natl. Acad. Sci. USA*, (in press).
31. Weber,J.L. and May,P.E. (1989) *Am. J. Hum.Genet.*, **44**, 388–396.