BIOINFORMATICS

Vol. 19 no. 3 2003, pages 362–367
DOI: 10.1093/bioinformatics/btf878

LTR_STRUC: a novel search and identification program for LTR retrotransposons

Eugene M. McCarthy* and John F. McDonald

Department of Genetics, University of Georgia, Athens, GA 30602, USA

Received on February 3, 2002; revised on May 20, 2002; July 11, 2002; accepted on August 2, 2002

ABSTRACT

Motivation: Long terminal repeat (LTR) retrotransposons constitute a substantial fraction of most eukaryotic genomes and are believed to have a significant impact on genome structure and function. Conventional methods used to search for LTR retrotransposons in genome databases are labor intensive. We present an efficient, reliable and automated method to identify and analyze members of this important class of transposable elements.

Results: We have developed a new data-mining program, LTR_STRUC (*LTR* retrotransposon *struc*ture program) which identifies and automatically analyzes LTR retrotransposons in genome databases by searching for structural features characteristic of such elements. LTR_STRUC has significant advantages over conventional search methods in the case of LTR retrotransposon families having low sequence homology to known queries or families with atypical structure (e.g. non-autonomous elements lacking canonical retroviral ORFs) and is thus a discovery tool that complements established methods. LTR_STRUC finds LTR retrotransposons using an algorithm that encompasses a number of tasks that would otherwise have to be initiated individually by the user. For each LTR retrotransposon found, LTR_STRUC automatically generates an analysis of a variety of structural features of biological interest.

Availability: The LTR_STRUC program is currently available as a console application free of charge to academic users from the authors.

Contact: gm@uga.edu; mcgene@uga.edu

INTRODUCTION

Retrotransposons are a major component of eukaryotic genomes. For example, at least 40 percent of the human genome is composed of retrotransposons (Yoder *et al.*, 1997). Long terminal repeat (LTR) retrotransposons have a structure and mode of replication similar to infectious retroviruses (Coffin *et al.*, 1997). All retrotransposons are distinguished by a life cycle involving an RNA intermediate. The RNA genome of a retroelement is copied into a double-stranded DNA molecule by reverse transcriptase and is subsequently integrated into the host genome.

We here describe a new data-mining program, LTR_STRUC (*LTR* retrotransposon *struc*ture) that provides a rapid, automated method for finding, delineating, and analyzing LTR retrotransposons in large genome databases. In identifying LTR retroelements in nucleotide sequence data, the search algorithm used by LTR_STRUC seeks certain generic structural features of such elements. The algorithm differs from conventional search methods, in which locating and identifying transposons depends on sequence similarity to previously identified elements. LTR_STRUC identifies full-length LTR retrotransposons independent of sequence homology and, as such, is complementary to conventional search methods.

SYSTEM AND METHODS

A search algorithm with a structural basis

In scanning genomic sequence data, LTR_STRUC employs a search algorithm that first identifies putative LTRs and then analyzes them further, looking for other defining features of LTR retrotransposons. The structure of a typical LTR retrotransposon is shown in Figure 1. Structural features important to LTR_STRUC include two sites critical to replication, the primer binding site (PBS) and polypurine tract (PPT), as well as, the presence of dinucleotides at the ends of each LTR (typically TG and CA). Particularly important are the direct or 'target-site' repeats (see Figure 1). When a LTR retroelement inserts itself into host DNA, a short (4–6 bp) segment of host DNA is replicated at the site of insertion ('target site repeat' or TSR). This feature allows LTR_STRUC to make an exact demarcation of the limits of a putative element.

Step 1: Finding An Initial Pair Of Matches: For each LTR retrotransposon present in the input file, LTR_STRUC first seeks the LTR pairs present at the ends of a putative element and then searches for additional characteristic retrotransposon features to confirm the hit.

*To whom correspondence should be addressed.

Published by Oxford University Press

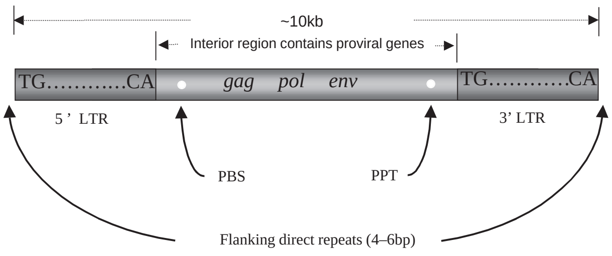
~10kb
Interior region contains proviral genes
TG..........CA
gag pol env
TG..........CA
5' LTR
3' LTR
PBS
PPT
Flanking direct repeats (4–6bp)

Fig. 1. Generic structure of an LTR retrotransposon. The typical retrotransposon encodes genes involved in the element's replication. The *pol* (*pol*ymerase) gene encodes reverse transcriptase/integrase proteins that are packaged within a capsid protein encoded by the *gag* (*g*roup specific *a*nti*g*en) gene. Reverse transcription is primed by a tRNA molecule binding to a site located 5′ to the *gag* gene called the 'primer binding site' (PBS). Infectious retroviruses and some LTR retrotransposons encode envelope proteins encoded by the *env* (*env*elope) gene. A 'polypurine tract' (PBS) is typically located just 5′ to the 3′ LTR. The LTR retrotransposon genome is bordered by long terminal repeats (LTRs). The ends of the LTRs typically end in the dinucleotides TG and CA.

The two LTRs of a particular element may fall anywhere within the contig, but the distance (D) between their 5′ ends should fall within the range dictated by the expected range of lengths characteristic of LTR retrotransposons (see below). Thus, it is possible to specify reasonable values for $d_{\min}$ and $d_{\max}$ such that the relationship $d_{\min} < D < d_{\max}$ will hold true for the vast majority of LTR retrotransposons (even those bearing large inserts).

Suppose (I):

(1) that the first nucleotide of a sequence S_i lies at position i in the input nucleotide sequence and denote the length of S_i by $L_S(i)$ (Figure 2);

(2) that both S_i and its two endpoints (the two nucleotides i and $i + L_S(i) - 1$) lie entirely within the bounds of an LTR, $L_{5'}$, situated at the 5′ end of an as yet unidentified LTR retrotransposon, R;

(3) that the first nucleotide of a second sequence M_k, which is highly similar to S_i, lies at position k of the input nucleotide sequence and denote the length of M_k by $L_M(k)$;

(4) that M_k and its two endpoints (the two nucleotides k and $k + L_M(k) - 1$) lie entirely within the bounds of $L_{3'}$, the 3′ LTR of R;

(5) that M_k lies at the same relative position within $L_{3'}$ as does S_i within $L_{5'}$;

If the conditions specified in (I) hold, then

$$i + d_{min} < k < i + d_{max}, \text{ where } d_{min} < D < d_{\max} \quad \text{(III)}$$

Thus, if for a given i, LTR_STRUC searches the interval $(i + d_{min}, i + d_{\max} + L_S(i) - 1)$ and finds a sequence M_k which is highly similar to S_i, then it has found a pair of sequences (S_i and M_k) such that one (S_i) is likely to lie in the 5′ LTR, and the other (M_k) is likely to lie in the 3′ LTR of a retrotransposon.

Now, further suppose (III):

(1) that the search of the input contig begins at its 5′ end;

(2) that the search proceeds in the 3′ direction taking sample sequences S_i at intervals of length Δi and, for each such sample sequence the search has scanned the interval, $i + d_{\min} < k < i + d_{\max} + L_S(i) - 1$, for a match M_k to S_i;

(3) that realistic values for d_{min} and d_{max} have been chosen.

If the conditions specified in (I) and (III) hold, then for any given i, the search for a match, M_k, to S_i has been exhaustive so long as the search has compared S_i to every sequence beginning with a nucleotide that falls in the range specified by (II). If, for any S_i, a sufficiently matching sequence, M_k, (that is one greater than 40 nucleotides in length and exhibiting greater than 70% homology) is found, such that the first nucleotide, k, of M_k lies in the interval $i + d_{\min} < k < i + d_{\max}$, then LTR_STRUC proceeds to Step 2 (alignment of putative LTRs). Otherwise i is incremented by Δi and a match for sequence $S_{i+\Delta i}$ is sought.

Step 2: Alignment of regions flanking the initial pair of matches: Alignment proceeds outward from the site of the initial match, (S_i and M_k). See Figure 2. Regions 3′ to S_i and M_k (i.e. e_1, e_2, e_3, e_4, and e_5 in Figure 2) are aligned first, then those 5′ direction (e_6 and e_7). The alignment proceeds outward by steps of size N. Examination of alignment results has shown that high quality alignments are obtained if $N = 100$.

The alignment steps outward because it is unknown at any given step whether the high levels of sequence similarity present in regions that have already been aligned will continue into adjacent regions. This approach to alignment saves computer time by avoiding alignment of regions outside the putative LTRs. It also provides a means of detecting their approximate ends since the expectation is that sequence similarity will fall to near random levels once the alignment passes the end of the LTRs.

The value of N sets an upper bound on the length of any given extension (See Figure 2). The extension is composed of the two sequences aligned in a single 'step' one from the putative 5′ LTR and one from the putative 3′ LTR. Figure 2 illustrates the alignment of two LTRs in seven successive extensions (e_1, e_2, e_3, e_4, e_5, e_6, e_7). The lengths of the two sequences composing an extension will usually differ.

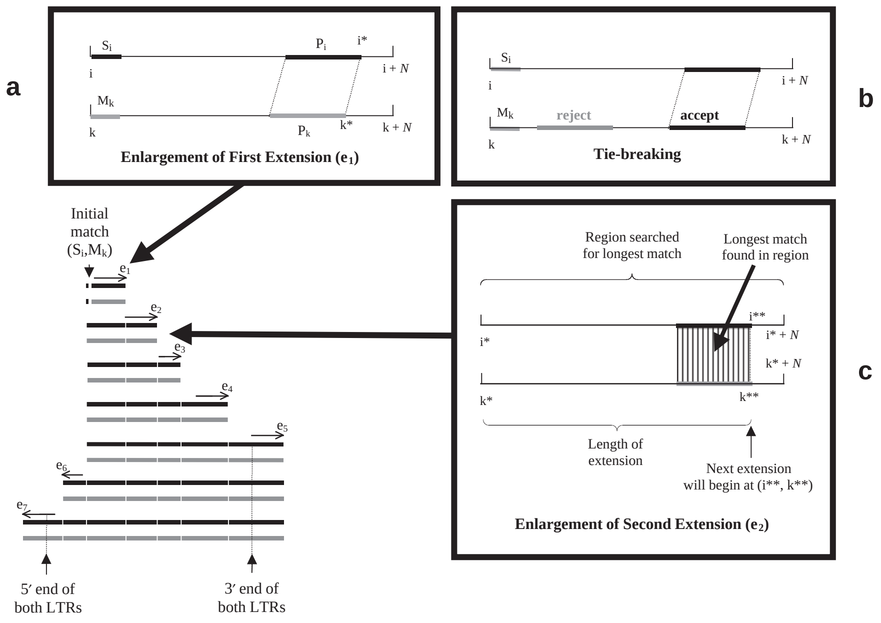

Fig. 2. The alignment of an LTR proceeds in steps: LTR_STRUC extends LTR alignments in steps (e_1, e_2, e_3, e_4, e_5, e_6, and e_7) beginning each extension by finding the largest match (P_i, P_k) in the next interval (2a) and (2c) and then aligning the region between that match and the portion of the alignment that has already been completed (see Figure 3). When two matches that are of equal quality and length are identified, LTR_STRUC breaks the tie by determining which of the two possible pairs lie most nearly 'opposite' each other in the alignment (2b). When LTR_STRUC reaches the 3′ end of the LTR it proceeds backwards from the start point to complete the alignment (steps represented in the figure by extensions, e_6 and e_7).

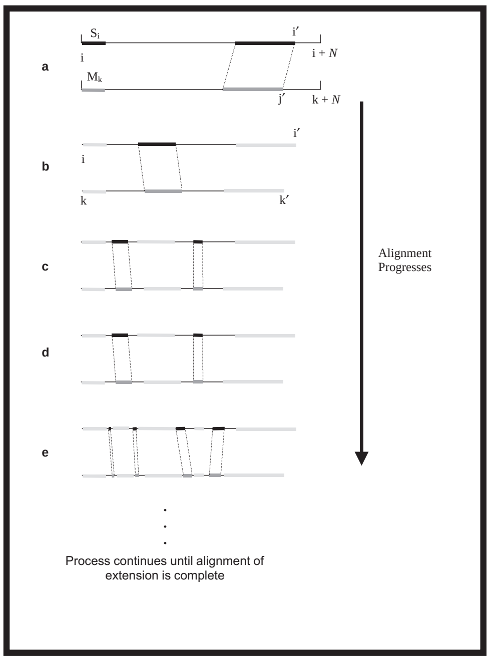

Fig. 3. LTR_STRUC uses a recursive process to complete the alignment of each extension: Once the endpoints of an extension are defined (3a), LTR_STRUC examines each remaining unaligned subinterval and then aligns the largest match in that subinterval. This process is repeated until all subintervals in the current extension are aligned (3b–e).

Starting with the first extension, extensions are added in the 3′ direction until the level of similarity between the aligned sequences falls below 70% for two successive extensions. Then extension begins in the 5′ direction and is continued until similarity between aligned sequences again falls below the 70% criterion for two successive extensions. At this point in the algorithm the LTRs are considered to be fully aligned and the analysis proceeds to the next step.

Step 3: Identification of the approximate endpoints of the aligned LTRs: Two adjacent windows β and γ, each of length 100 bp, are passed across the LTR alignment. These windows may be described as the intervals $(j - 100, j - 1)$ and $(j, j + 99)$, respectively, where j is an index specifying position in the alignment. As j is incremented and the windows pass across the alignment, at each position, T_β (the total number of matches in window β) is subtracted from T_γ (the total number of matches in window γ) to obtain the difference

$$\Delta \quad = T\gamma - T_\beta.$$

Where Δ reaches a maximum and minimum, k will be a good approximation of the respective positions in the alignment of the 5′ and 3′ ends of the LTRs; the former of these two values of Δ denote $\hat{j}_{5'}$ and the latter, $\hat{j}_{3'}$.

Step 4: Determination of exact end points: The two alignment positions, $\hat{j}_{5'}$ and $\hat{j}_{3'}$, generated by Step 3

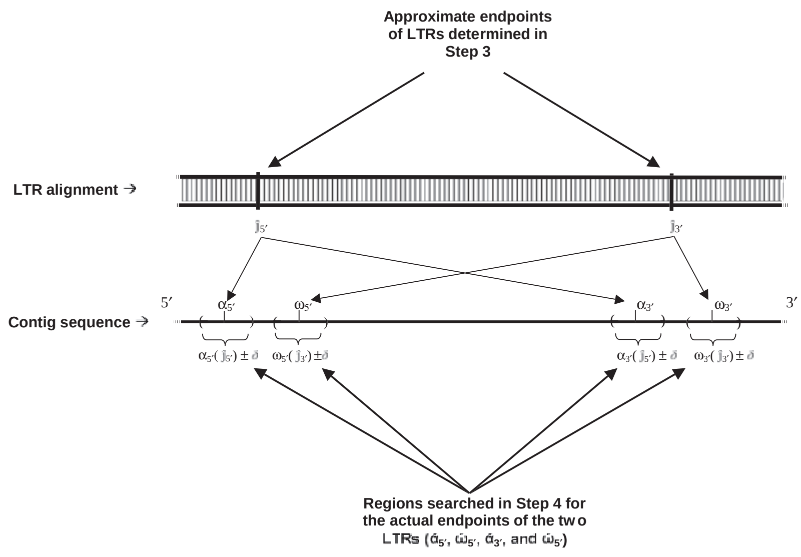
Approximate endpoints of LTRs determined in Step 3
LTR alignment →
$\hat{j}_{5'}$
$\hat{j}_{3'}$
5′
3′
Contig sequence →
$\alpha_{5'}$
$\omega_{5'}$
$\alpha_{3'}$
$\omega_{3'}$
$\alpha_{5'}(\hat{j}_{5'}) \pm \delta$
$\omega_{5'}(\hat{j}_{3'}) \pm \delta$
$\alpha_{3'}(\hat{j}_{5'}) \pm \delta$
$\omega_{3'}(\hat{j}_{3'}) \pm \delta$
Regions searched in Step 4 for the actual endpoints of the two LTRs ($\acute{\alpha}_{5'}$, $\acute{\omega}_{5'}$, $\acute{\alpha}_{3'}$, and $\acute{\omega}_{5'}$)

Fig. 4. Endpoints of the LTRs are first approximated in the alignment and then analyzed further: In Step 3 (see text) estimated alignment positions ($\hat{j}_{5'}$ and $\hat{j}_{3'}$) of the endpoints are obtained for the LTRs. In Step 4 (see text) these positions are translated into a quartet of estimated endpoints in the contig (i.e. $\alpha_{5'}(\hat{j}_{5'})$, $\omega_{5'}(\hat{j}_{3'})$, $\alpha_{3'}(\hat{j}_{5'})$, and $\omega_{3'}(\hat{j}_{3'})$). LTR_STRUC then scores all possible endpoint combinations in the vicinity of this quartet in order to find the actual endpoints.

correspond to four indices in the original input contig, that is, to a pair of approximate endpoints for the 5′ LTR, $\alpha_{5'}(\hat{j}_{5'})$, $\omega_{5'}(\hat{j}_{3'})$, and an equivalent pair for the 3′ LTR, $\alpha_{3'}(\hat{j}_{5'})$, and $\omega_{3'}(\hat{j}_{3'})$. (Figure 4). Since these indices are good approximations of the actual endpoints, denoted $\acute{\alpha}_{5'}$, $\acute{\omega}_{5'}$, $\acute{\alpha}_{3'}$, and $\acute{\omega}_{5'}$, of the two LTRs, the inequalities

$$|\acute{\alpha}_{5'} - \alpha_{5'}(\hat{j}_{5'})| < \delta, \quad |\acute{\omega}_{5'} - \omega_{5'}(\hat{j}_{3'})| < \delta,$$
$$|\acute{\alpha}_{3'} - \alpha_{3'}(\hat{j}_{5'})| < \delta, \text{ and } |\acute{\omega}_{5'} - \omega_{3'}(\hat{j}_{3'})| < \delta, \quad \text{(IV)}$$

should hold for some small integer δ (Figure 4).

The score given each of the quartets, ($\alpha_{5'}$, $\omega_{5'}$, $\alpha_{3'}$, $\omega_{3'}$), not only determines which quartet will be chosen as the endpoints of the posited transposon (i.e. the quartet receiving the maximum score), but also serves as a measure of how likely it is that the posited LTR retrotransposon is real. In practice we have found that an 'element' for which the maximum quartet score is low is usually not an LTR retrotransposon at all. As the maximum quartet score approaches one (the highest possible score), however, it becomes a near certainty that the hit actually does represent an LTR retrotransposon. Using these scores to rank-order the output data in terms of hit quality has significantly reduced the amount of labor required in subsequent analysis of results.

IMPLEMENTATION

The LTR_STRUC is written in Visual C++ (Microsoft version 6.0) and runs on PC platforms. The program is available from the authors as a console application. LTR_STRUC reads nucleotide sequence files in FASTA format. LTR_STRUC breaks sequences containing strings of separator symbols (such as '*n*', '*N*', or '−') at the point where the separator string occurs, treating the sequences on either side of the string as separate contigs. Input files must be downloaded and scanned locally on the user's computer. The user can specify certain parameters:

(1) maximum and minimum overall length of the transposon;
(2) maximum and minimum lengths for the LTRs;
(3) cutoff score.

LTR_STRUC generates report files (in text format) only for hits generating a score in excess of the cutoff score. These files contain a detailed analysis of each hit. They include all the information enumerated in Table 1.

RESULTS AND DISCUSSION

The search algorithm used by LTR_STRUC consists of four steps: (1) location of an initial pair of matches (S_i, M_k), which might lie within an LTR pair (these matches need not be exact); (2) alignment of the regions adjacent to the initial match; (3) identification of the approximate end points of the putative LTRs; and (4) determination of exact end points of the putative LTRs. In addition, LTR_STRUC's reporting function provides analytic output of specific interest to researchers studying retrotransposons (Table 1).

Table 1. Information in LTR_STRUC output files

Name of source contig
Location of element within contig
Score for current hit
Lengths of contig, element, LTRs, and largest ORF
Nucleotide sequences for the whole transposon, TSRs
LTRs, PBS, PPT, dinucleotides terminating the LTRs
Orientation of the transposon within the contig
(determined by relative positions of PBS and PPT)
Sequences for all ORFs (longer than 50 amino acids)
Intra-element percent identify of LTRs
An alignment of the putative LTRs

The conventional approach to identifying LTR retrotransposons in genome databases typically begins by scanning input nucleotide sequences for sequences showing similarity to known reverse transcriptases (RTs). From the standpoint of identifying LTR retrotransposons, this approach has three inherent drawbacks: (1) it biases the search toward elements containing RTs that are similar to the query RT; (2) it overlooks LTR elements that lack a reverse transcriptase — such elements are common in some species (Witte *et al.*, 2001); (3) even after an RT has been identified by the traditional method, completion of the analysis typically requires a number of additional steps for each putative element found. The first of the two negative aspects of the conventional approach are significant problems for researchers wishing to find *all* LTR retrotransposons in a given input data set. The third means extra labor that can be avoided by using LTR_STRUC because it eliminates a series of additional steps required in the conventional approach.

LTR_STRUC is only effective for full-length LTR retrotransposons. To date, our tests indicate that its sensitivity and accuracy in *locating* elements in this category are comparable to established techniques. However, LTR_STRUC has been found to be superior to existing techniques with regard to *identifying* LTR retrotransposons that belong to families having novel structure or that have low sequence homology to previously recognized families. To find new families of LTR retrotransposons, a researcher has to sift through a typically lengthy list of anonymous sequences and analyze their structures before any definite identification can be made. In contrast, because LTR_STRUC relies on an algorithm that focuses on a structural analysis of the input nucleotide sequence, any hit with a high score provides the user with a high degree of assurance that the hit in question actually is an LTR retrotransposon. From the standpoint of discovering new families, BLAST searches have the inherent drawback that all of the high-score hits will be similar to the known query. Prior to the development of LTR_STRUC, our laboratory conducted a search for the presence of LTR retrotransposons in the *C. elegans* database (www.wormbase.org) using the RT from the *Cer1* elements (acc# U15406) as a query sequence. After several months of analysis, workers in our lab were able to identify and characterize 12 families of *C. elegans* LTR retrotransposons (Bowen and McDonald 1999). As a test of the accuracy and reliability of LTR_STRUC, we utilized LTR_STRUC to search the same database for LTR retrotransposons. Within two hours, LTR_STRUC had identified and characterized all 12 families of *C. elegans* LTR retrotransposons identified in our previous study, as well as, three additional elements not detected in our initial analysis.

We have also employed LTR_STRUC to search for LTR retrotransposons in the rice (*Oryza sativa)* genome (McCarthy *et al.*, 2002). In that survey, LTR_STRUC identified 32 new families of LTR retrotransposons, increasing the number of known, named LTR retrotransposon families in the rice genome to a total 59. Among the novel families of LTR retrotransposons identified in this study were four non-autonomous families having no significant homology to any previously described retrotransposon or retroviral proteins. These elements were detected through their structural characteristics and could easily have been overlooked by conventional search methods.

Most of the transposons detected by LTR_STRUC have had LTRs that are more than 90% identical. The vast majority of full-length LTR retrotransposons found by conventional methods, also, show high levels of LTR identity, again, usually in excess of 90% (e.g. Promislow *et al.*, 1999; Bowen and McDonald, 1999, 2000). Occasionally, in our surveys of the rice genome LTR_STRUC detected elements with LTR–LTR similarities falling into the 80% range, but these, for the most part, were elements bearing regional duplications that extend unbroken for tens of base pairs, not ones that are heavily peppered with single-nucleotide mutations. Given the set-up of the algorithm, LTR_STRUC is unlikely to detect elements where levels of LTR identity fall below about 75%.

LTR_STRUC can find nested insertions of LTR retrotransposons. LTR_STRUC found examples of such nesting during the scan of the rice genome. However, to detect instances where one retroelement inserts into another, the size of the search window (i.e. $d_{\max} - d_{\min}$) must be increased so that it will exceed the size of the retrotransposon receiving the insertion (which will make the program run more slowly). LTR_STRUC will also find LTR retrotransposons bearing other types of insertions (so long as the LTR pair and TSRs are still recognizable).

Because it looks first for the LTRs at opposite ends of an element, LTR_STRUC cannot find elements that span contigs. The entire element (plus some flanking sequence) must be present in a single contig for LTR_STRUC to find it. For the same reason, LTR_STRUC will not locate truncated elements or solo LTRs. To date, in surveying a particular genome, our method of identifying truncated copies in a given family has been first to identify as many families as possible either, in the literature or using LTR_STRUC. We then conduct BLAST searches against available sequence data using as queries one full-length copy from each of those families. LTR_STRUC is not intended to find truncated retrotransposons directly. Its main utility is in the discovery and initial analysis of new *families* of retrotransposons.

From the user's perspective, the program rapidly generates information that otherwise requires a great deal labor. Program run times show a linear relation with the length of the scanned contig length, but the time required increases with the square of the search window size ($d_{\max} - d_{\min}$) and maximum LTR length. Experience in our lab has shown that the overall task of finding new families of retrotransposons in genomic sequence data is significantly facilitated by LTR_STRUC. We estimate that the reduction in user time is on the order of 100–1000-fold (hours vs. months).

REFERENCES

Bowen,N. and McDonald,J.F. (1999) Genomic analysis of *Caenorhabditis elegans* reveals ancient families of retroviral-like elements. *Genome Res.*, **9**, 924–935.

Bowen,N. and McDonald,J.F. (2000) *Drosophila* euchromatic LTR retrotransposons are much younger than the host species in which they reside. *Genome Res.*, **11**, 1527–1540.

Coffin,J.M., Hughes,S.H. and Varmus,H.E. (1997) *Retroviruses*, Plainview, Coldspring Harbor Laboratory Press, NY.

McCarthy,E.M., Liu,J., Gao,L. and McDonald,J.F. (2002) LTR Retrotransposons of *Oryza sativa*. *Genome Biol.*, **3**, research 0053.1–0053.11.

Promislow,D.E., Jordan,I.K. and McDonald,J.F. (1999) Genomic demography, a life history analysis of transposable element evolution. *Proc. Roy. Soc. Lon. B Biol.*, **266**, 1555–1560.

Witte,C.P., Le,Q.H., Bureau,T. and Kumar,A. (2001) Terminal-repeat retrotransposons in miniature (TRIM) are involved in restructuring plant genomes. *PNAS, USA*, **98**, 13778–13783.

Yoder,J.A., Walsh,C.P. and Bestor,T.H. (1997) Cytosine methylation and the ecology of intragenomic parasites. *Trends Genet.*, **13**, 335–340.