REPuter: the manifold applications of repeat analysis on a genomic scale

Stefan Kurtz*, Jomuna V. Choudhuri, Enno Ohlebusch, Chris Schleiermacher[1], Jens Stoye[2] and Robert Giegerich

Faculty of Technology, University of Bielefeld, PO Box 10 01 31, D-33501 Bielefeld, Germany, [1]Artemis Pharmaceuticals, Neurather Ring 1, 51063 Köln, Germany and [2]Max Planck Institute for Molecular Genetics, Department for Computational Molecular Biology, Ihnestrasse 73, 14195 Berlin, Germany

Received July 25, 2001; Revised and Accepted September 19, 2001

ABSTRACT

The repetitive structure of genomic DNA holds many secrets to be discovered. A systematic study of repetitive DNA on a genomic or inter-genomic scale requires extensive algorithmic support. The *REPuter* program described herein was designed to serve as a fundamental tool in such studies. Efficient and complete detection of various types of repeats is provided together with an evaluation of significance and interactive visualization. This article circumscribes the wide scope of repeat analysis using applications in five different areas of sequence analysis: checking fragment assemblies, searching for low copy repeats, finding unique sequences, comparing gene structures and mapping of cDNA/EST sequences.

INTRODUCTION

One of the most striking features of DNA is the extent to which it consists of repeated substrings. This is particularly true of eukaryotes. For example, it is estimated that families of reiterated sequences account for >50% of the human genome (1). The presence of palindromic (i.e. reverse complemented) repeats hints to the formation of hairpin structures that may provide some structural or replicational mechanism (2). Furthermore, some repeats have been shown to affect bacterial virulence by acting as the molecular basis of a mechanism used to successfully colonize and infect different human individuals (3). This makes repeats an interesting research topic, and indeed there is a vast literature on repetitive structures and their hypothesized functional and evolutionary role.

The manifold applications of repeat analysis

The obvious task of a computer program for automatic repeat analysis is to find in, for example, a complete genome, all repeats above a given level of significance. Such analysis alone can give interesting insights into the structure of the genome like an abnormal distribution of repetitive elements or recent duplication events. Usually, the repeats will then be further investigated by other methods, typically starting with a database search for any known characteristics of the newly detected repeat sequence.

Besides presenting a program for this traditional type of repeat analysis, the emphasis of this article is on the fact that an efficient and complete computation of repeats is a powerful algorithmic technique in a variety of tasks normally not subsumed under repeat analysis. In fact, of the five applications that we present in Applications, only the one dealing with low copy repeats related to human malformations reports on the traditional kind of repeat analysis. The other applications that we consider go far beyond this: checking sequence assemblies; finding unique sequences as a preprocessing step of PCR primer or DNA oligo chip design; comparing gene structures in order to verify gene intron/exon predictions; and mapping ESTs to genomic sequence.

This versatility (yet to be described) results from the fact that a repeat is a mathematically simple object—a substring w of a sequence S occurring twice in S (allowing a certain amount of error). This has two consequences: a substring w that is not a repeat in S is unique in S; a common substring of sequences S_1 and S_2 (also allowing a certain amount of error) is a repeat of the concatenated sequence S_1S_2. With these two simple observations it is immediately clear that repeat analysis can be used for the various types of sequence analysis tasks mentioned above, once it has been rigorously implemented so as to be applicable to whole genome sequences.

The challenge of repeat analysis on a genomic scale

A tool for the systematic study of the repetitive structure of sequences as large as complete genomes must satisfy the following criteria. (i) Efficiency: to analyze complete genomes, up to 3–4 billion bp, the space and time used by the algorithm must scale practically linear with the sequence length. (ii) Flexibility and significance: a biologically realistic model must recognize not only exact, but also degenerate, not only direct, but also palindromic repeats. To determine the significance of a repeat, a statistical assessment is mandatory. (iii) Interactive visualization: the large amount of data generated requires an overview of the whole input sequence, but the user must also be able to zoom in on details of particular repetitive regions. (iv) Compositionality: as repeat finding is considered to be a basic step in genome structure analysis, the program

*To whom correspondence should be addressed. Tel: +49 521 106 2906; Fax: +49 521 106 6411; Email: kurtz@techfak.uni-bielefeld.de must provide a simple interface to enable composition with other advanced analysis tools.

The *REPuter* program described herein satisfies these requirements in the following way. The search engine *REPfind* of *REPuter* uses an efficient and compact implementation of suffix trees in order to locate exact repeats in linear space and time. It has been estimated in Kurtz (4) that this time-critical task can be done in linear time for sequences up to the size of the human genome. These exact repeats are used as seeds from which significant degenerate repeats are constructed allowing for mismatches, insertions and deletions. Yet, our program is not heuristic: it guarantees to find all degenerate repeats according to the parameters specified by the user—minimum length and maximum number of errors. The output is sorted by significance scores (*E*-values). Besides degenerate direct repeats, *REPfind* is capable of detecting degenerate palindromic repeats. One of the basic innovations of *REPfind* is the ability to process very large DNA sequences very fast. For example, *REPfind* computes all 18 391 maximal degenerate repeats with a length of at least 700 bp with at most 10 errors in the complete genome of *Drosophila melanogaster* at a speed of 100 000 bases per second. More running time and space results are reported in Kurtz *et al.* (5) and in Applications.

The output of the search engine *REPfind* is displayed in the form of a repeat graph by the interactive visualization program *REPvis*. This feature is described in detail in Interactive Visualization.

The stand-alone version of *REPuter* described here is available free of charge for non-commercial purposes. See http://www.genomes.de for more details. An online version of *REPuter* providing some basic functionality can be used on the Bielefeld Bioinformatics web server (http://bibiserv.techfak.uni-bielefeld.de/reputer/). There are currently more than 50 laboratories using the stand-alone version, and more than 340 submissions to the online version per month.

Related work

There are many papers describing algorithms for finding repeats in a string [see Kurtz *et al.* (5) for an overview]. Here we concentrate on available software tools.

Two types of approaches to locate repeats in biological sequences can be distinguished. Methods of the first type, whose most prominent example is *RepeatMasker* (http://ftp.genome.washington.edu/RM/RepeatMasker.html; A.F.A.Smit and P.Green, unpublished), define a ‘repeat’ as a substring that is known to occur very often in a genome. Such substrings tend to confuse sequence analysis programs, and hence they are masked to avoid spurious results. Repeat masking programs use a dictionary of known repeat sequences and perform an exact or approximate string matching of the given sequence against all the dictionary entries.

Programs of the other type, like *REPuter*, try to find repeats without prior knowledge, just from the nucleic acid sequence. The simplest approach to such *a priori* repeat detection is to look at a dot plot of the sequence against itself. Here, repeats show up as diagonal lines. Such dot plots can, for example, be produced by the programs *Dotter* [http://www.cgr.ki.se/cgr/groups/sonnhammer/Dotter.html; (6)] or *Large Dot Plots* (http://alces.med.umn.edu/rawdot.html; Virtual Genome Center, unpublished).

For the algorithmic detection of repeats several methods have been suggested, and any suite of biosequence analysis programs contains one or more repeat finding tools, like *repeat* in the GCG package [http://www.accelrys.com/products/gcg_wisconsin_package/; (7)] or *etandem*, *equicktandem*, *einverted* and *palindrome* in the EMBOSS package [http://www.hgmp.mrc.ac.uk/Software/EMBOSS/; (8)]. However, the algorithms behind these programs are heuristic and not very well described in the literature or the program documentation.

Several stand-alone programs also exist for finding repeats. *Dst* (http://alces.med.umn.edu/newdst.html; Virtual Genome Center, unpublished) is a heuristic method based on filtration of fixed-length oligonucleotides. *REPRO* [http://mathbio.nimr.mrc.ac.uk/~rgeorge/repro/; (9)] finds repeats in protein sequences. The method is based on computation of non-overlapping local alignments, and hence uses time proportional to the square of the sequence length. This naturally restricts the search to short sequences. *OligoRep* (http://wwwmgs.bionet.nsc.ru/mgs/programs/oligorep/; unpublished) is in spirit similar to our method, but is restricted to short sequences (a maximal length of 1000 bp is recommended). A step further in repeat analysis is to cluster the repeats according to their position, for example, as described in Volfovsky *et al.* (10).

A very important subtype of repeats are tandem repeats where the two (or more) copies of the repeat immediately follow each other in the DNA sequence. Programs specialized in finding tandem repeats include *Tandem Repeat Finder* [http://c3.biomath.mssm.edu/trf.html; (11)] and *Tandyman* (http://www.stdgen.lanl.gov/tandyman/index.html; unpublished).

In all this work either the methods are restricted to small input or they do not implement the full model of degenerate repeats. *REPuter* provides the first solution to efficient and exhaustive repeat analysis of complete genomes. Thus, it is the only tool useful for applications beyond repeat analysis considered here.

MODELS AND ALGORITHMS

Those readers who are less interested in technical details, but more in the applications of the *REPuter* program, can safely skip this section upon first reading.

Basic notions

Let S be a string of length $|S| = n$ over an alphabet Σ. $S[i]$ denotes the ith character of S, for $i \in [1, n]$. S^{-1} denotes the reverse of S. For $i \leq j$, $S[i, j]$ denotes the substring of S starting with the ith and ending with the jth character of S. Substring $S[i, j]$ is denoted by the pair of positions (i, j). The length of the substring (i, j) is $l(i, j) = j - i + 1$. To refer to the characters to the left and right of every character in S without worrying about the first and last character, we define $S[0]$ and $S[n + 1]$ to be two distinct characters not occurring anywhere else in S.

A pair of positions (i_1, j_1), $i_1 \leq j_1$, contains a pair (i_2, j_2), $i_2 \leq j_2$, if and only if $i_1 \leq i_2$ and $j_2 \leq j_1$. A pair (p_1, p_2) of substrings (i.e. a pair of pairs of positions) contains a pair (p_3, p_4) of substrings if and only if p_1 contains p_3, and p_2 contains p_4.

A pair of substrings $R = ((i_1, j_1),(i_2, j_2))$ is an exact repeat if and only if $(i_1, j_1) \neq (i_2, j_2)$ and $S[i_1, j_1] = S[i_2, j_2]$. The length of

R is $l(R) = j_1 - i_1 + 1 = j_2 - i_2 + 1$. An exact repeat $R = ((i_1, j_1),(i_2, j_2))$ is called 'maximal' if and only if $S[i_1 - 1] \neq S[i_2 - 1]$ and $S[j_1 + 1] \neq S[j_2 + 1]$ [see Gusfield (12)].

If S is a DNA sequence, then we distinguish between two kinds of biologically interesting repeats. The repeats defined above are called direct (or forward) repeats. A pair of substrings $P = ((i_1, j_1),(i_2, j_2))$ is a 'palindromic' (or 'reverse complemented') repeat if and only if $S[i_1, j_1] = \overline{S[i_2, j_2]}$, where $\overline{w}$ denotes the reverse complement of a DNA sequence w.

The Hamming distance of two equal-length strings S_1 and S_2, denoted by $d_H(S_1, S_2)$, is the number of positions where S_1 and S_2 differ.

There are three kinds of edit operations: *deletions*, *insertions* and *mismatches* of single characters. These are collectively referred to as differences. The edit distance or Levenshtein distance of S_1 and S_2, denoted by $d_E(S_1, S_2)$, is the minimum number of edit operations needed to transform S_1 into S_2.

Finding exact repeats

The standard approach to compute maximal exact repeats is based on hashing methods. These usually tabulate for each DNA sequence w of a fixed length, say r, the positions $P(w)$ in S, where w occurs. For each w and $i, j \in P(w)$, $i < j$, $((i, i + r - 1),(j, j + r - 1))$ is an exact repeat of length r. To find maximal repeats of length at least l, each exact repeat has to be extended to the left and to the right to check if it is embedded in a maximal exact repeat of the required length. The extension is done by pairwise character comparisons, and thus the running time does not only depend on the number of maximal repeats, but also on their lengths. In practice, r is between 10 and 13, a restriction imposed by the hashing techniques. On the other hand, l is usually of length at least 20. Hence there are many more repeats to be extended than maximal repeats to be reported. To exemplify this, we have calculated the corresponding numbers for the genome of the *Escherichia coli* K12 strain (13). This genome is of length 4 639 221. There are 4 105 188 repeats of length 12 to be extended, but only 7799 maximal repeats of length at least 20. Thus, the ratio of maximal repeats against tested candidates is 0.0019.

Using the suffix tree for S we do not have to filter out a small number of maximal repeats from many candidates. Instead, we directly compute maximal exact repeats (of arbitrary length), using the algorithm of Gusfield (12). This algorithm runs in $O(n + z)$ time, where z is the number of maximal repeats. Thus, the running time is optimal and does not depend on the lengths of the repeats. It has independently been shown how to practically construct suffix trees for genomic-size sequences (4,14). The space efficient implementation techniques developed by Kurtz (4), and an efficient implementation of the algorithm of Gusfield (12), were the basis of the first *REPuter* program for finding exact repeats (15). This subtask of our new algorithms is not discussed further.

We will present algorithms for finding degenerate repeats based on two different distance models: the Hamming distance model and the edit distance model. In the following we assume that an error threshold $k \geq 0$ and a length threshold $l > 0$ are given.

The mismatches repeat problem

k-mismatch repeats are based on the notion of Hamming distance. A pair of equal-length substrings $R = ((i_1, j_1),(i_2, j_2))$ is a k-mismatch repeat if and only if $(i_1, j_1) \neq (i_2, j_2)$ and $d_H(S[i_1, j_1], S[i_2, j_2]) = k$. The length of R is $l(R) = j_1 - i_1 + 1 = j_2 - i_2 + 1$. A k-mismatch repeat is maximal covering if it is not contained in any other k-mismatch repeat.

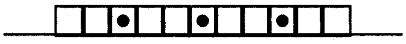


Figure 1. $k = 3$ mismatching characters (denoted by filled circles) distributed equally over a repeat of length $l = 11$, yielding a minimum seed size of $\left\lfloor \frac{l}{k+1} \right\rfloor = \left\lfloor \frac{11}{4} \right\rfloor = 2$.

The 'mismatches repeat problem' is to enumerate all maximal covering k-mismatch repeats of length at least l that occur in S. Our algorithm maximal mismatch repeats (MMR) for solving this problem is based on the following observation. If $R = ((i_1, j_1),(i_2, j_2))$ is a k-mismatch repeat, then the k mismatches divide $S[i_1, j_1]$ and $S[i_2, j_2]$ into maximal exact repeats $w_0, w_1, w_2, \ldots, w_k$. The exact repeats w_0 and w_k occurring at the borders of the strings are maximal because R is maximal covering; the others are obviously maximal. Now $\max_{i \in [0,k]} |w_i|$ is a minimum if the mismatching character pairs are equally distributed over R, yielding a pattern as shown in Figure 1. Obviously, for such an equal distribution the length of the longest w_i is $\geq \left\lceil \frac{l-k}{k+1} \right\rceil = \left\lfloor \frac{l}{k+1} \right\rfloor$. Therefore, every maximal covering k-mismatch repeat R of length l contains a maximal exact repeat of length $\geq \left\lfloor \frac{l}{k+1} \right\rfloor$, called a seed.

Algorithm MMR. Compute all seeds and test for each seed whether it can be extended to a k-mismatch repeat. More precisely, for each seed $((i_1, j_1),(i_2, j_2))$ tables H_{left} and H_{right} of size $k + 1$ are computed such that for each $q \in [0, k]$: $H_{right}(q)$ is the maximum number p such that $d_H(S[j_1 + 1, j_1 + p], S[j_2 + 1, j_2 + p]) = q$, i.e. $S[j_1 + 1, j_1 + p]$ and $S[j_2 + 1, j_2 + p]$ have an alignment with q mismatches. Analogously, $H_{left}(q)$ is the maximum number p such that $d_H(S[i_1 - p, i_1 - 1], S[i_2 - p, i_2 - 1]) = q$. Then, for each $q \in [0, k]$, check whether $j_1 - i_1 + 1 + H_{left}(q) + H_{right}(k - q) \geq l$ holds. If so, output the maximal covering k-mismatch repeat $((i_1 - H_{left}(q), j_1 + H_{right}(k - q)),(i_2 - H_{left}(q), j_2 + H_{right}(k - q)))$.

It is easy to prove that algorithm MMR correctly solves the mismatches repeat problem.

Table H_{right} can be computed in $O(k)$ time by using a suffix tree that allows to determine the length of the longest common prefix of two substrings of S in constant time. Since we construct the suffix tree of S anyway, this imposes virtually no overhead. Of course, the same approach can be applied to H_{left} [For details on this technique see Harel and Tarjan (16) and Schieber and Vishkin (17)].

The overall time efficiency of algorithm MMR can be assessed as follows. The preprocessing phase of computing the suffix tree and locating the seeds takes $O(n)$ time. For a given seed, the extension phase of algorithm MMR takes $O(k)$ time as shown above, yielding an overall time efficiency of $O(n + zk)$ where z is the number of seeds. The number of seeds z can be estimated by $E[z] = O\left(n^2 \frac{1}{|\Sigma|^s}\right)$ where $|\Sigma|$ is the size of the alphabet Σ and $s = \left\lfloor \frac{l}{k+1} \right\rfloor$ (see below).

Algorithm MMR detects a maximal k-mismatch repeat more than once if it contains more than one seed. This can be avoided by stopping the computation of table H_{left} as soon as another seed is detected. This ensures that for a given seed the

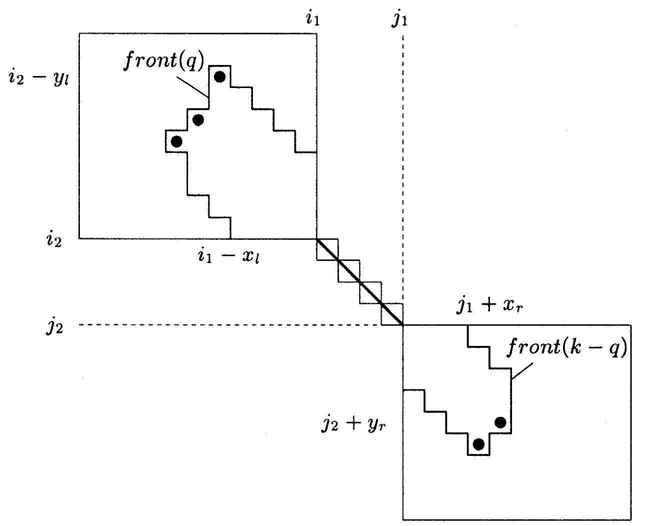


Figure 2. Extension of a seed in algorithm MDR. The elements of $E_{\mathrm{left}}(q)$ and $E_{\mathrm{right}}(k - q)$ are marked by filled circles.

algorithm will output only those maximal k-mismatch repeats in which this particular seed is the leftmost.

The differences repeat problem

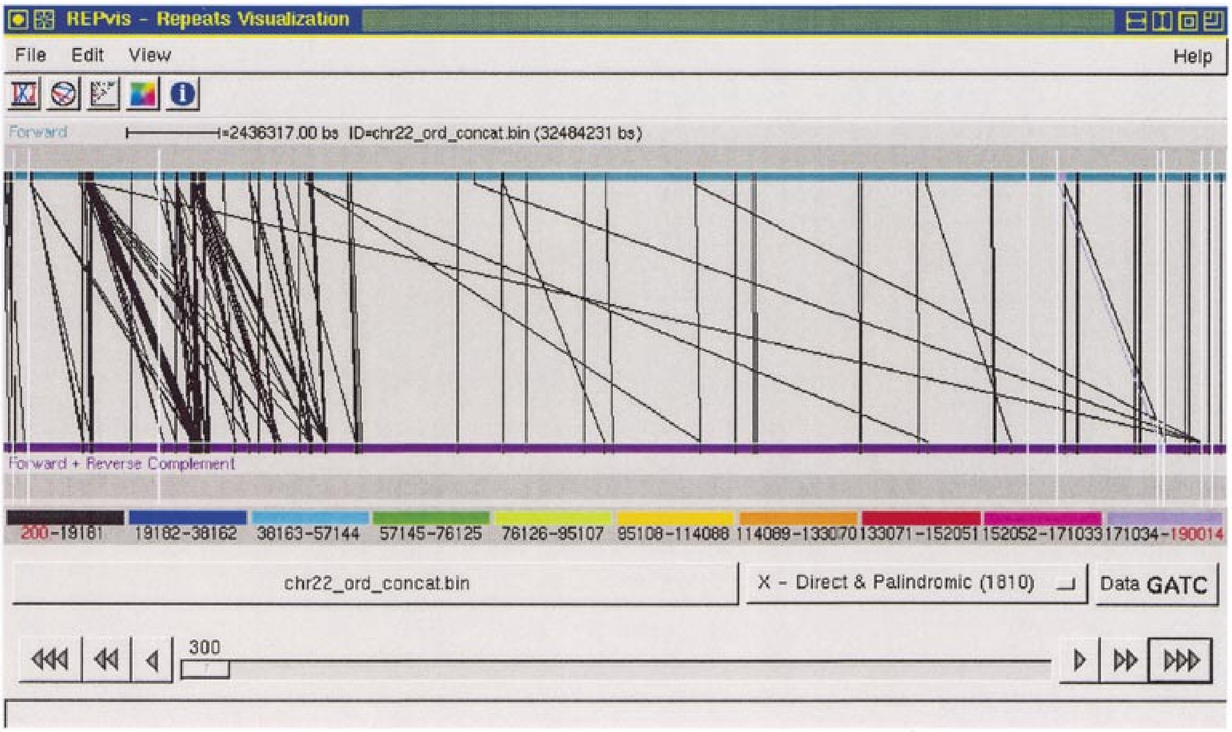
We now extend our technique to allow for insertions and deletions. A pair $R = ((i_1, j_1),(i_2, j_2))$ of substrings is a k-differences repeat if and only if $(i_1, j_1) \neq (i_2, j_2)$ and $d_E(S[i_1, j_1], S[i_2, j_2]) = k$. The 'length' of R is $l(R) = \min(j_1 - i_1 + 1, j_2 - i_2 + 1)$. A k-differences repeat is 'maximal covering' if it is not contained in any other k-differences repeat.

The differences repeat problem is to enumerate all maximal covering k-differences repeats of length at least l. Our algorithm maximal differences repeats (MDR) (for solving this problem also crucially depends on the fact that every maximal covering k-differences repeat R of length l contains a maximal exact repeat of length $\geq \left\lfloor \frac{l}{k+1} \right\rfloor$, called a seed.

Algorithm MDR. Compute all seeds and try to extend these to k-differences repeats as shown in Figure 2. To be more precise, for every seed $((i_1, j_1),(i_2, j_2))$ compute tables E_{left} and E_{right} defined as follows: $E_{\mathrm{right}}(q)$ is the set of all pairs $(x_r, y_r) \in [1, m] \times [1, n]$ such that $d_E(S[j_1 + 1, j_1 + x_r], S[j_2 + 1, j_2 + y_r]) = q$, i.e. $S[j_1 + 1, j_1 + x_r]$ and $S[j_2 + 1, j_2 + y_r]$ have an alignment with q differences. Analogously, $E_{\mathrm{left}}(q)$ is the set of all pairs $(x_l, y_l) \in [1, m] \times [1, n]$ such that $d_E(S[i_1 - x_l, i_1 - 1]^{-1}, S[i_2 - y_l, i_2 - 1]^{-1}) = d_E(S[i_1 - x_l, i_1 - 1], S[i_2 - y_l, i_2 - 1]) = q$. For each $q \in [0, k]$, for each pair $(x_l, y_l) \in E_{\mathrm{left}}(q)$, and each $(x_r, y_r) \in E_{\mathrm{right}}(k - q)$: if $j_1 - i_1 + 1 + x_l + x_r \geq l$ and $j_2 - i_2 + 1 + y_l + y_r \geq l$, then output the maximal covering k-differences repeats $((i_1 - x_l, j_1 + x_r), (i_2 - y_l, j_2 + y_r))$.

It is not difficult to show that algorithm MDR correctly solves the differences repeat problem.

One could of course use a standard dynamic programming (DP) algorithm [for example see Wagner and Fischer (18)] to extend seeds in $O(n^2)$ time. However, there are faster methods: using the algorithm of Ukkonen (19), it is possible to compute



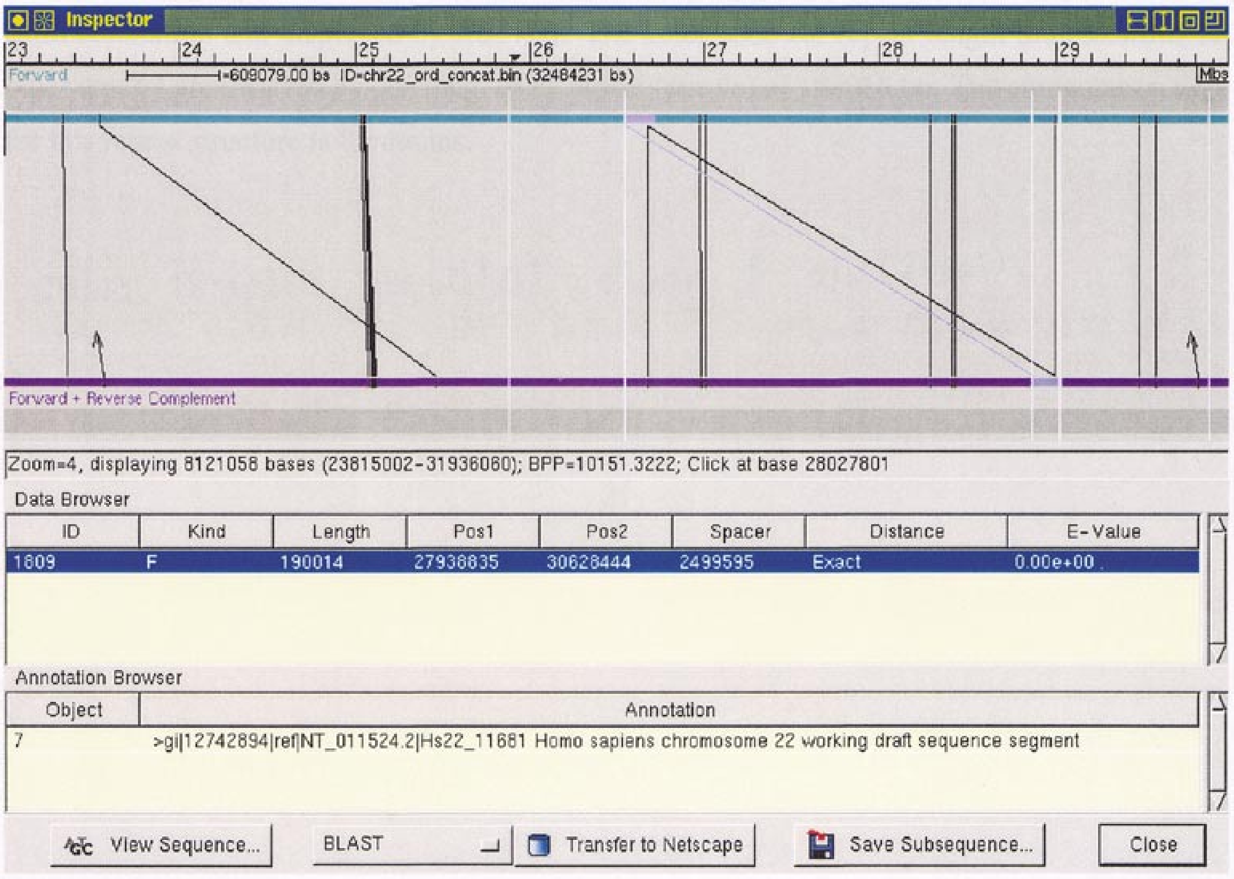
=2436317.00 bs ID=chr22_ord_concat.bin (32484231 bs)

chr22_ord_concat.bin

X - Direct & Palindromic (1810)

300

Figure 3. Assembly checking of human chromosome 22. *REPvis* display of exact direct and palindromic repeats with a minimum length of 300 bp. The chromosome (32 484 231 bp) consists of 11 concatenated contigs separated by vertical white lines (the separators are specified as an extra annotation and displayed by *REPvis*). The color code for repeat length indicates that all other repeats are dwarfed by one long, exact repeat (light purple) of 190 014 bp. The computation time for this repeat structure is 8 min.



Zoom=4, displaying 8121058 bases (23815002-31936060); BPP=10151.3222; Click at base 28027801

ID	Kind	Length	Pos1	Pos2	Spacer	Distance	E-Value
1809	F	190014	27938835	30628444	2499595	Exact	0.00e+00

Object	Annotation
7	>gi|12742894|ref|NT_011524.2|Hs22_11681 Homo sapiens chromosome 22 working draft sequence segment

Figure 4. Assembly checking of human chromosome 22 (continued). Enlarged view of contig 7 and contig 8, with attention to the 190 014 bp repeat, shown in light purple. Obviously, the entire contig 8 has also incorrectly been assembled at the beginning of contig 7. This error is rectified in the current release of human chromosome 22.

tables E_{left} and E_{right} in $O(kn)$ time by computing only *front*(k) of the DP-matrix, see Figure 2. A combination of this algorithm with the longest common prefix technique yields an $O(k^2)$ time method to compute tables E_{left} and E_{right}. Lastly, for each $q \in [0, k]$ the algorithm tests $O(k^2)$ combinations of the values in $E_{\text{left}}(q)$ and $E_{\text{right}}(k - q)$. Thus, the extension phase of algorithm MDR takes time $O(k^3)$ per seed. As for algorithm MMR, we conclude that the overall time efficiency of algorithm MDR is $O(n + zk^3)$, where z is the number of seeds.

By restricting to left-most seeds, algorithm MDR can be improved in a similar way to algorithm MMR.

A different approach to search for degenerate repeats would be to initially search for inexact seeds and then to extend these with less errors. However, this approach suffers from the fact that there is no efficient algorithm for finding all inexact seeds, even if the number of errors is very small.

Significance of repeats

In order to assess the significance of a repeat found by our methods, we compute its E-value, i.e. the number of repeats of the same length or longer and with the same number of errors or fewer that one would expect to find in a random DNA of the same length.

As a model of random DNA the uniform Bernoulli model is used, where each base A, C, G and T has the same probability $p = 1/4$ of occurrence. (In general, $p = 1/|\Sigma|$ for an alphabet Σ of size $|\Sigma|$). The E-value of the number of maximal exact repeats of length $\geq l$ can be computed according to the following formula [for details see Kurtz *et al.* (5)]:

$$E[\text{\# of maximal exact repeats of length } \geq l]$$

$$= \frac{1}{2}(n - l + 1)(n - l)p^l(1 - p) + (n - l)p^{l+1}$$

Therefore,

$$E[\text{\# of maximal exact repeats of length } \geq l] \in O(n^2 p^l)$$

E-values for k-mismatch repeats can be computed in a similar way. The probability that two independent sequences S_1 and S_2, both of length l, have a Hamming distance of exactly k under the uniform Bernoulli model is

$$Pr[d_H(S_1, S_2) = k] = \binom{l}{k} p^{l-k}(1 - p)^k .$$

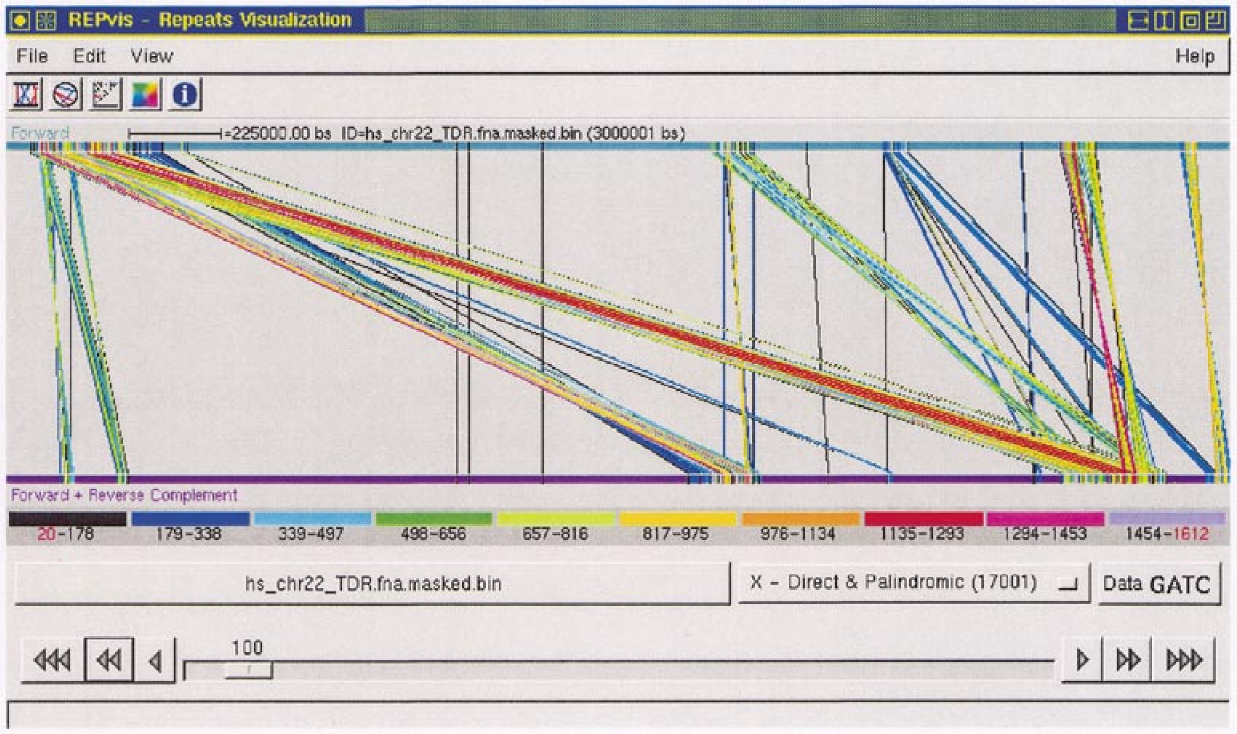
Using this formula, E[# of maximal $\leq k$-mismatch repeats of length $\geq l$] can be approximated by

$$\frac{1}{2}n(n - 1)\binom{l}{k} p^{l-k}(1 - p)^{k+2}$$

The exact solution can be found in Kurtz *et al.* (5).

One can generalize this result for the non-uniform Bernoulli model with fixed probabilities p_a for each $a \in \Sigma$. This is achieved by replacing p by the probability p^* that the bases at two randomly chosen positions of S are the same,

$$p^* = \sum_{a \in \Sigma} (p_a)^2.$$



=225000.00 bs ID=hs_chr22_TDR.fna.masked.bin (3000001 bs)

hs_chr22_TDR.fna.masked.bin

X - Direct & Palindromic (17001)

Figure 5. Low copy repeats. The repeat graph of the selected 3 Mb from human chromosome 22 showing an unusual repeat region (see left part of Fig. 3). The direct and palindromic repeats of minimum length 100 bp are shown. In essence, the net-like pattern reveals a low copy repeat, with some direct repeats, and others reversed. Comparing this with the repeat pattern observed in (23), one observes a correspondence to the 3 Mb (TDR) of chromosome 22q11.2, responsible for the DiGeorge/Velo-cardio-facial syndrome. The computation time for this repeat structure is 30 s.

This, however, is only an approximation to the exact solution because the different probabilities for self-overlapping repeats are ignored.

In the case of the edit distance no analytic solution is known for $Pr[d_E(S_1, S_2) = k]$. For this reason we use the procedure of Kurtz and Myers (20), which estimates the probability of the event $A_k(P)$ that an arbitrary (not necessarily random) string P matches the prefix of a random string with edit distance k. This procedure is an unbiased estimator which gives good results in a matter of 1000 samples even for patterns of small probability. To obtain an estimation $Pr[d_E(S_1, S_2) = k]$, we precomputed a table E. Here $E(l, k)$ is the average of the estimation of the probability of the event $A_k(P)$. The estimation is delivered by running the above procedure with 1000 samples for 100 random patterns P, each of length l. The variance of the 100 estimations obtained for each l and k is very small, and so we argue that $E(l, k)$ gives a good approximation for $Pr[d_E(S_1, S_2) = k]$ where $l = \max(|S_1|, |S_2|)$. Hence E[# of maximal $\leq k$-differences repeats of length $\geq l$] can be approximated by $\frac{1}{2}n(n-1)E(l, k)$

INTERACTIVE VISUALIZATION

REPvis, the visualization component of the *REPuter* program, provides an easy to use interface for examining repeat structures computed by *REPfind*. The program is designed to be used by the biologist, thus putting the data in the hands of those who can best interpret it.

A typical mode of use is as follows. The visualization comes up showing a single colored line, depicting either the longest or the most significant repeat. The first step is to obtain an impression of the overall number and distribution of repeats. By shifting a slider, we let further repeats rise on the screen, in the order of decreasing length or significance, which is coded in a 10-color scale (see Figs 3, 5, 6, 8 and 9). Since black is used as the color for the shortest/least significant repeats, we may go down all the way: if we hit the noise level (see Figs 6 and 7) the more significant repeats still shine up in colors before a black background of noise.

During the overview, we may catch interest in particular repeats or repeat-rich regions. A mouse click brings up the inspection window (see Figs 4 and 7). Here we can zoom in or out on a region by left or right clicking the mouse. Selecting a position on the strand symbol prints the information corresponding to this sequence position in a browser box below. There, a single repeat can be selected to view the alignment of the two instances of the repeat, or to submit the corresponding nucleotide sequence for further investigation of biological significance to a FASTA or BLAST database search.

The repeat graph as displayed by *Repvis* can be annotated with additional symbols or lines. For example, the user can display a predicted gene structure by specifying colored arcs and their position in the genome. These will be shown together with the repeat graph, e.g. to generate or verify hypotheses about the correspondence between the particular repeat structure and an intron/exon structure (also see Applications).

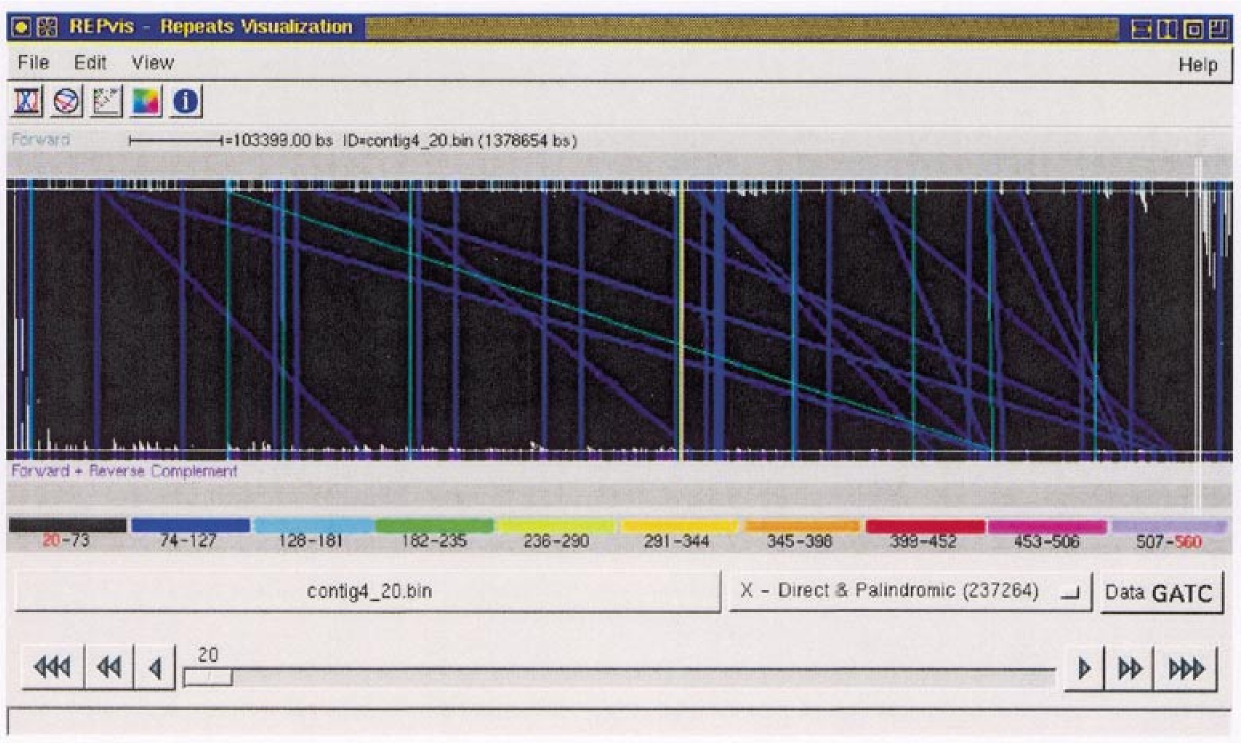

REPvis - Repeats Visualization
File
Edit
View
Help
Forward
1=103399.00 bs ID=contig4_20.bin (1378654 bs)
Forward + Reverse Complement
20-73
74-127
128-181
182-235
236-290
291-344
345-398
399-452
453-506
507-560
contig4_20.bin
X - Direct & Palindromic (237264)
Data GATC
20

Figure 6. Unique sequences. Direct and palindromic repeats of contig 4 of human chromosome 21 with a minimum length of 20 bp and up to 2 errors. LINEs and SINEs were masked. Clearly, the noise level of random repeats is reached, generating a black background. The computation time for this repeat structure is 16 s. This example is continued in Figure 7.

APPLICATIONS

The basic application of *REPuter* is, of course, to reveal the repetitive structure of large chromosomes or genomes. Our recent experience shows that the use of *REPuter* extends far beyond the original purpose. This is illustrated by applications in five different sequence analysis tasks. The running time of *REPfind* for each application is given in the figure legends. It refers to a 400 MHz SUN computer with the Solaris 2.5.1. operating system.

Assembly checking

The task of assembly programs is to arrange sequence reads in the proper order and orientation to obtain complete BAC sequences and contigs. However, assembly programs are not perfect and the assembly is often erroneous, possibly resulting in a poor annotation of the resulting sequence.

An assembled sequence can be checked by applying *REPfind* to it. Very long repeats may be due to overlapping regions between BAC sequences or contigs. This may hint to assembly errors. If repeats are palindromic, one of the repeat instances may have been assembled in the wrong orientation.

This procedure was applied to the 11 concatenated contigs of human chromosome 22. The contigs were obtained from GenBank on March 22, 2001 (accession nos NT_011516.3, NT_011517.2, NT_011519.4, NT_025937.1, NT_011520.5, NT_011521.1, NT_011523.4, NT_011524.2, NT_011525.3, NT_019197.2, NT_011526.3). The *REPvis* repeat graph (Fig. 3) indicates an overlapping region of unexpected length. Zooming into the repeat graph (Fig. 4) reveals that the complete contig 8 has already been assembled in the beginning of contig 7. This error has been corrected in the current version of human chromosome 22.

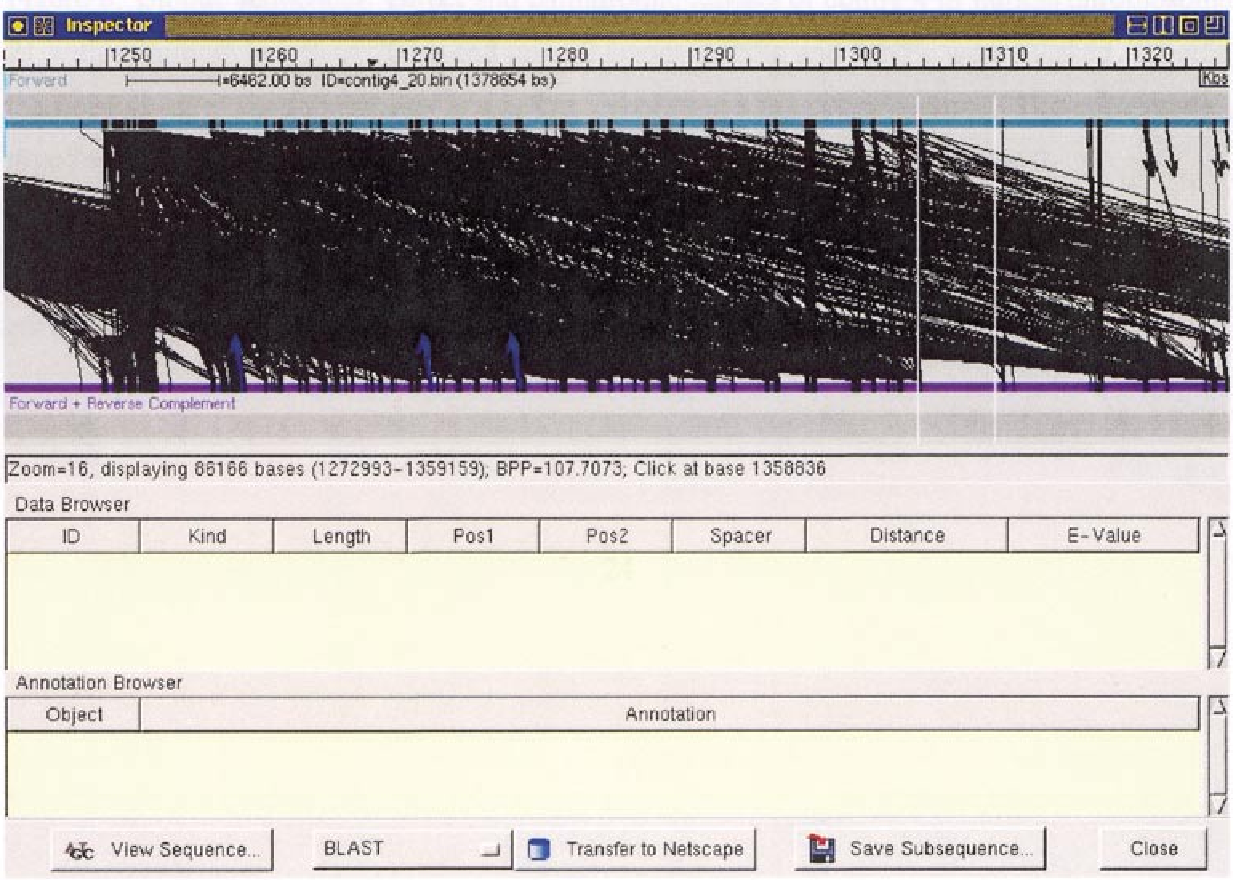
Low copy repeats related to human malformations

Human malformations and syndromes are often associated with the deletion or duplication of specific chromosomal regions. It has been observed that phenotypes related to deletions are more severe than those related to duplications (21). Chromosomal regions rich in gene content may be directly associated with several malignant diseases caused by microdeletions in this part of the genome. The haploinsufficiency for at least some of the genes in the deleted region may be responsible for direct effects on specific developmental processes.

One well known aberration associated with such chromosomal rearrangements is the DiGeorge/Velo-cardio-facial syndrome, localized at 22q11.2 (22,23). Most of the rearrangements observed in this region refer to large deletions in a 3 Mb region, causing various anomalies, including mental retardation. Some specific low copy repeat elements are identified flanking this typical deleted region (TDR). This suggests that they function as breakpoints leading to homologous recombination, explaining the genomic instability of this chromosome region.

Low copy repeats form a characteristic net-like pattern in the visualization provided by *REPuter*. At a glance, this pattern indicates the number of repeats, their relative positions and the regions potentially deleted.

We used *Repfind* to locate direct and palindromic repeats for human chromosome 22 (Fig. 3). In the left part of the repeat graph, an agglomeration of repeats is observed covering a range of ~3 Mb containing many repeated blocks (Fig. 5). The



Zoom=16, displaying 86166 bases (1272993-1359159); BPP=107.7073; Click at base 1358836

ID	Kind	Length	Pos1	Pos2	Spacer	Distance	E-Value

Object	Annotation

Figure 7. Unique sequences (continued). Zooming into the repeat graph of Figure 6, we find that there are still regions free from repeats. The area between the two vertical white lines represents 5300 bp. It provides unique candidates for primer design.

pattern involves one essential element that was repeated four times, sometimes directly, sometimes reversed. A comparison with experimental results of Shaikh *et al.* (23) suggests that the structure corresponds to the TDR responsible for the DiGeorge/Velo-cardio-facial syndrome.

Unique sequences

Hybridization techniques are very effective tools in molecular biology. The research pursued spans a wide range including genotyping, pre-natal diagnostics and microarray technology.

Hybridization techniques make use of nucleic acid probes to detect complementary nucleic acid targets present in biological fluids or tissues. Their success essentially depends on the specificity of the probe. Targeting the probes to non-unique sequences may result in cross-hybridization generating false positives. Finding unique sequences is the mathematical complement of finding repeats. Since *REPuter* solves the repeat finding problem in a non-heuristic way, it can also be used to solve the complementary problem. Assume the probes should have length l and be unique up to k errors. Determining and discarding all repeats of minimum length l and a maximum of k errors, the remaining sequence fragments are guaranteed to be unique everywhere. Hence, probes can be designed from them according to other experimental criteria.

Our example application is the screening of BAC libraries to get clones used as probes for fluorescence *in situ* hybridization in contig 4 of human chromosome 21 (GenBank accession no. NT_003534). As this screening is done by PCR, our strategy was to filter out all direct and palindromic repeats with a minimum length of 20 bp, allowing 2 errors (Figs 6 and 7). This guarantees the absence of mispriming (in contig 4) for all primers chosen from the unique sequences. After this preprocessing step, primers can be designed separately for each region of interest, using standard software.

Gene structure comparison

Comparative genomics is one of the major reasons for sequencing whole genomes of different organisms. Throughout evolution, vital genes and regulatory regions have been conserved to guarantee basic functions, maintaining similarities across species. Since the mouse is a well known model organism for studies of human biology and medicine, the access to its genome allows researchers to make important discoveries in the regulation of human genes based on common structures and mechanisms.

REPuter enables us to compare two or more sequences by concatenating them, and then searching for repeats. Controlling the allowed error rate in repeats, we can consider many grades of similarity. In this way, *REPuter* provides a simple plausibility check for gene structures predicted by other software tools.

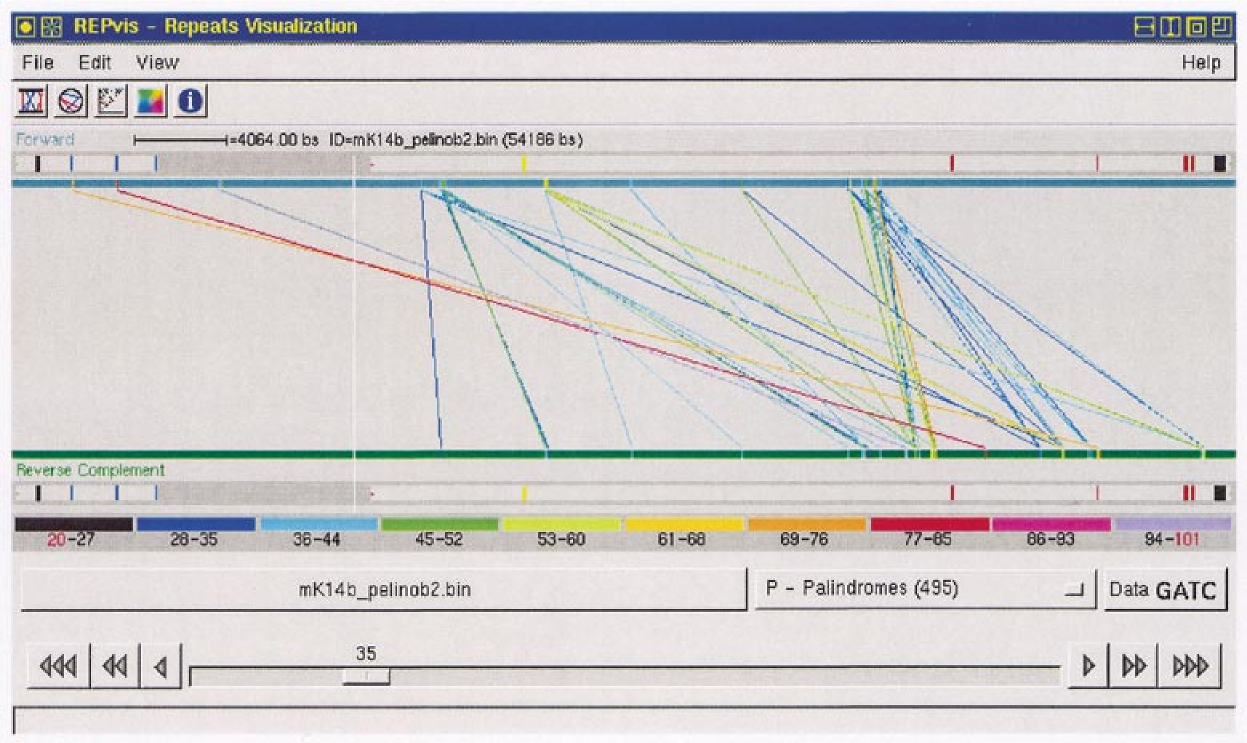


Figure 8. Gene structure comparison. The concatenated 5′-UTR from mouse and human pellino1 gene is considered. Both sequences are separated by the vertical white line. Since the pellino gene is expressed in opposed strands in mouse and human, we restrict to palindromic repeats. The repeat graph shows palindromic repeats of 35 bp minimum length with at most two errors. The Genscan prediction is shown as an annotation of the repeat graph, each exon being represented by a colored bar. By analyzing only the inter-sequence matches, we can identify similarities and conserved regions between mouse and human with respect to the pellino gene. The purple line suggests that a common exon is missed by Genscan. The computation time for this repeat structure is 3 s.

In our application example, we consider the 5′-UTR region from the *Mus musculus* pellino1 gene (GenBank accession no. AC091421) and the complete pellino1 gene of *Homo sapiens* (GenBank accession no. NT_005326.1; Fig. 8). The gene structure predicted by the program Genscan [version 1.0, with default options; (24)] is shown with the repeat graph. Two of the predicted mouse exons coincide with matches in the human genomic sequence. However, there is yet another 98% identity match between both species, 101 bp long (purple). This region was not predicted as an exon by Genscan, although the strong conservation and a match with a cDNA (GenBank accession nos AF302503 for *Mouse musculus* and AF302505 for *Homo sapiens*) suggest that it is indeed an exon.

cDNA/EST mapping

Expressed sequence tags (ESTs) are markers that provide a rapid and reliable method for gene discovery as well as a resource to analyze the expression of known and unknown genes (25).

Given a collection of ESTs and a genomic sequence, one wants to find local similarities between them. This can be done by concatenating the genome with all ESTs and searching for repeats.

We applied this strategy to look for a mouse gene similar to the human *Kiaa0903* gene. There are known homologous regions between human chromosome 2 holding the *Kiaa0903* gene and mouse chromosome 11. The human cDNA *Kiaa0903* sequence was concatenated with a contig from the mouse region (GenBank accession no. AC091423; Fig. 9). The repeat graph shows that the exons of the *Kiaa0903* gene are separated by very long introns. The 3′- and 5′-ends of the *Kiaa0903* gene are either missing in the contig or are not well conserved.

CONCLUSIONS

We have given five examples for applications of automatic repeat analysis. On a small scale, all of these analyses can be done by *ad-hoc* combinations of traditional tools. However, all-against-all comparisons break down when data size approaches that of a small eukaryotic genome or a human chromosome. Heuristic search methods are inadequate when the presence or absence of repetitive elements must be determined with certainty. The virtue of *REPuter* is to combine linear time efficiency with exhaustive analysis. Thus, a single tool can be utilized for all analysis tasks which are directed to, or can be formulated as, tasks of repetitive structure analysis.

ACKNOWLEDGEMENTS

S.K. was partially supported by grant KU 1257/1 from the Deutsche Forschungsgemeinschaft. J.V.C. was supported by the DFG-Graduiertenkolleg 635 Bioinformatik.

REFERENCES

1. Lander,E.S., Linton,L.M., Birren,B., Nusbaum,C., Zody,M.C., Baldwin,J., Devon,K., Dewar,K., Doyle,M., FitzHugh,W. *et al.* (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

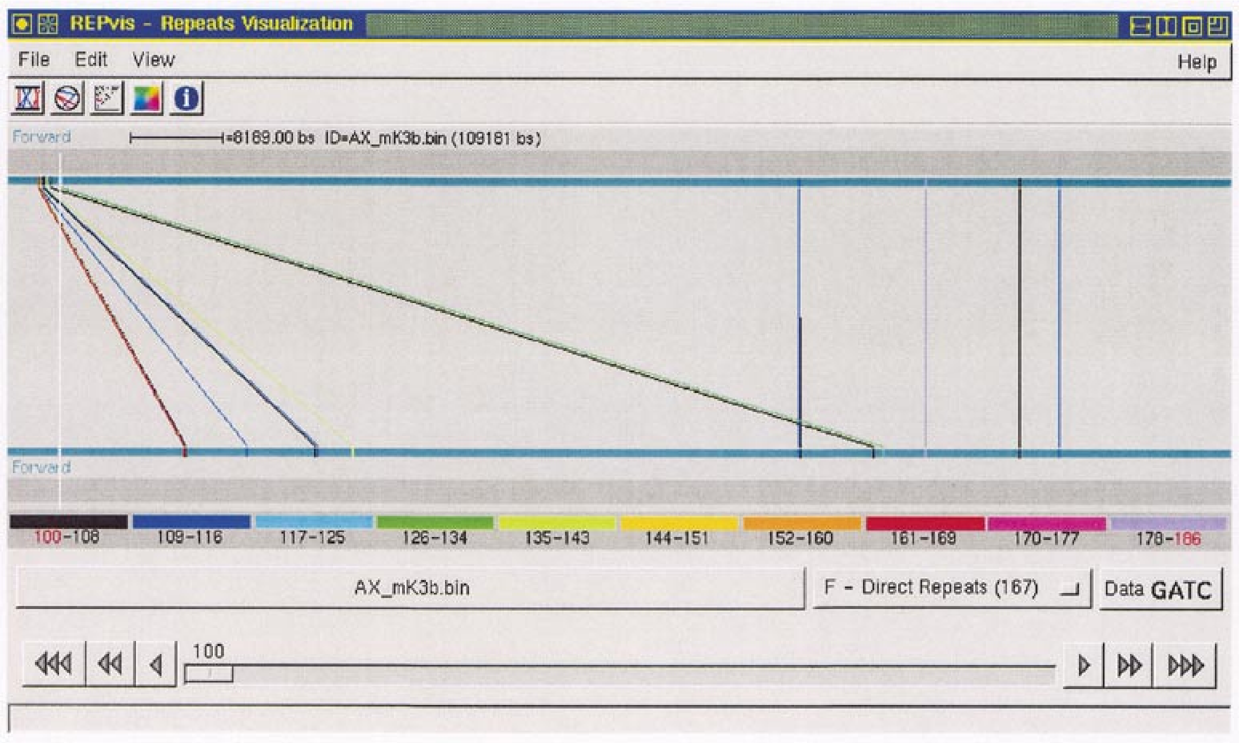


Figure 9. cDNA/EST mapping. Looking for a mouse gene similar to the human *Kiaa0903* gene. There are known homologous regions between human chromosome 2 holding the *Kiaa0903* gene and mouse chromosome 11. The human cDNA *Kiaa0903* sequence was concatenated with a 104 270 bp contig from the mouse region. Both sequences are separated by a vertical white line. The repeat graph shows that the exons of the *Kiaa0903* gene are separated by very long introns. The 3′- and 5′-ends of the *Kiaa0903* gene are either missing in the contig or are not well conserved. The computation time for this repeat structure is 3 s.

2. Huang,C., Lin,Y., Yang,Y., Huang,S. and Chen,C. (1998) The telomeres of *Streptomyces* chromosomes contain conserved palindromic sequences with potential to form complex secondary structures. *Mol. Microbiol.*, **28**, 905–916.
3. van Belkum,A., Scherer,S., van Alphen,L. and Verbrugh,H. (1998) Short sequence DNA repeats in prokaryotic genomes. *Microbiol. Mol. Biol. Rev.*, **62**, 275–293.
4. Kurtz,S. (1999) Reducing the space requirement of suffix trees. *Software—Practice and Experience*, **29**, 1149–1171.
5. Kurtz,S., Ohlebusch,E., Schleiermacher,C., Stoye,J. and Giegerich,R. (2000) Computation and visualization of degenerate repeats in complete genomes. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Menlo Park, CA, pp. 228–238.
6. Sonnhammer,E.L.L. and Durbin,R. (1995) A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. *Gene*, **167**, GC1–GC10.
7. Womble,D.D. (1999) GCG : the Wisconsin package of sequence analysis programs. In Misener,S. and Krawetz,S.A. (eds), *Bioinformatics Methods and Protocols*, *Methods in Molecular Biology.* Humana Press, Totowa, NJ, **132**, pp. 3–22.
8. Rice,P., Longden,I. and Bleasby,A. (2000) EMBOSS : the European Molecular Biology Open Software Suite. *Trends Genet.*, **14**, 473–475.
9. George,R.A. and Heringa,J. (2000) The REPRO server: finding protein internal sequence repeats through the web. *Trends Biochem. Sci.*, **25**, 515–517.
10. Volfovsky,N., Haas,B. and Salzberg,S. (2001) A clustering method for repeat analysis in DNA sequences. *Genome Biol.*, **2**, research0027.1–0027.11.
11. Benson,G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res.*, **27**, 573–580.
12. Gusfield,D. (1997) *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, NY.
13. Blattner,F., Plunkett,G., Bloch,C., Perna,N., Burland,V., Riley,M., Collado-Vides,J., Glasner,J., Rode,C., Mayhew,G., Gregor,J., Davis,N., Kirkpatrick,H., Goeden,M., Rose,D., Mau,B. and Shao,Y. (1997) The complete genome sequence of *Escherichia coli* K-12. *Science*, **277**, 1453–1474.
14. Delcher,A., Kasif,S., Fleischmann,R., Peterson,J., White,O. and Salzberg,S. (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369–2376.
15. Kurtz,S. and Schleiermacher,C. (1999) REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics*, **15**, 426–427.
16. Harel,D. and Tarjan,R. (1984) Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, **13**, 338–355.
17. Schieber,B. and Vishkin,U. (1988) On Finding lowest common ancestors. *SIAM Journal on Computing*, **17**, 1253–1263.
18. Wagner,R. and Fischer,M. (1974) The string to string correction problem. *Journal of the ACM*, **21**, 168–173.
19. Ukkonen,E. (1985) Algorithms for approximate string matching. *Information and Control*, **64**, 100–118.
20. Kurtz,S. and Myers,G. (1997) Estimating the probability of approximate matches. In *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM'97)*. Springer Verlag, Berlin, pp. 52–64.
21. Brewer,C., Holloway,S., Zawalnyski,P., Schinzel,A. and FitzPatrick,D. (1998) A chromosomal deletion map of human malformations. *Am. J. Hum. Genet.*, **63**, 1153–1159.
22. Edelmann,L., Pandita,R. and Morrow,B. (1999) Low-copy-repeats mediate the common 3-Mb deletion in patients with velo-cardio-facial syndrome. *Am. J. Hum. Genet.*, **64**, 1076–1086.
23. Shaikh,T., Kurahashi,H., Saitta,S., O'Hare,A., Hu,P., Roe,B., Driscoll,D., McDonald-McGinn,D., Zackai,E., Budarf,M. and Emanuel,B. (2000) Chromosome 22-specific low copy repeats and the 22q11.2 deletion syndrome: genomic organization and deletion endpoint analysis. *Hum. Mol. Genet.*, **9**, 489–501.
24. Burge,C. and Karlin,S. (1997) Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.*, **268**, 78–94.
25. Adams,M.D., Kelley,J.M., Gocayne,J.D., Dubnick,M., Polymeropoulos,M.H., Xiao,H., Merril,C.R., Wu,A., Olde,B., Moreno,R.F., Kerlavage,A., McCombie,W. and Venter,J. (1991) Complementary DNA sequencing: expressed sequence tags and human genome project. *Science*, **252**, 1651–1656.