Visualising the repeat structure of genomic sequences

Nava E Whiteford [*]
School of Chemistry, University of Southampton, SO17 1BJ, Southampton, UK

Niall J Haslam [†]
Gerald Weber[‡]
School of Chemistry, University of Southampton, SO17 1BJ, Southampton, UK
Adam Prügel-Bennett
School of Electronics and Computer Science,
University of Southampton, SO17 1BJ, Southampton, UK
Jonathan W Essex
School of Chemistry, University of Southampton, SO17 1BJ, Southampton, UK
Cameron Neylon
School of Chemistry, University of Southampton, SO17 1BJ, Southampton, UK
and
ISIS Pulsed Neutron and Muon Source,
Rutherford Appleton Laboratory, Chilton, Didcot OX11 0QX, UK

Repeats are a common feature of genomic sequences and much remains to be understood of their origin and structure. The identification of repeated strings in genomic sequences is therefore of importance for a variety of applications in biology.

In this paper a new method for finding all repeats and visualising them in a two dimensional plot is presented. The method is first applied to a set of constructed sequences in order to develop a comparative framework. Several complete genomes are then analysed, including the whole human genome.

The technique reveals the complex repeat structure of genomic sequences. In particular, interesting differences in the repeat character of the coding and non-coding regions of bacterial genomes are noted.

The method allows fast identification of all repeats and easy intergenome comparison. In doing this the plot effectively creates a signature of a sequence which allows some classes of repeat present in a sequence to be identified by simple visual inspection.

To our knowledge this is the first time all exact repeats have been visualised in a single plot that highlights the degree to which repeats occur within a genomic sequence, giving an indication of the important

[*] Present address: School of Chemistry, Trinity College Dublin, College Green, Dublin 2, Ireland

[†] Present address: EMBL Heidelberg, Meyerhofstraße 1, 69117 Heidelberg, Germany

[‡] Present address: Department of Physics, University of Ouro Preto, 35.400-000 Ouro Preto-MG, Brazil

role repeats play. From this it is clear that large scale repeat analysis remains an important and unsolved problem in Bioinformatics.

1. Introduction

The repetition of both large and small sequences is a common feature of both eukaryotic and prokaryotic genomes [1, 2, 3], with some authors suggesting that as much as 50% of the human genome is composed of repetitions [4]. The biological role of repeated sequences has been investigated by a number of authors, where they have been linked to evolutionary mechanisms in prokaryotic organisms [5]. In the case of triplet repeats they have been linked to thermodynamic stability and the effect of this in genetic expansion diseases [6]. Tsuge *et al.* also recently reported on an association between three tandem repeats in the regulatory region of SMYD3 and human carcinogenesis [7], and detailed analysis of single sequence repeats in humans has been carried out by Subramanian *et al.* [8].

Several methods have been developed for analysing the repeat structure of genomic sequences [9, 10, 11, 12]. Most methods scan for a specific type of repeat such as short sequence repeats [1], palindromic repeats [13], tandem repeats [14, 15, 16, 17], or highly periodic short repeat elements [18, 19]. Usually, such methods are unable to detect repeats that do not match a predefined pattern and intra- or inter-genomic analyses are usually very difficult. Some methods, such as the use of Fourier transforms for repeat identification [18], do not search for a specific repeat pattern but rather try to locate occurrences of highly correlated periodic repeats. However this approach typically only identifies very strong genome-wide correlations such as those due to the triplet nature of the genetic code. The problems of identification combined with the size of large genomes makes identifying the full range of repeated sequences in genomes a challenging computational problem.

Computationally the use of suffix structures for genomic sequence analysis, which include suffix tree [20, 2, 21] and suffix array methods [22, 23], has greatly increased the efficiency of searching and storing strings. The analysis described here makes use of the suffix array [24], and the associated LCP [25] (Longest Common Prefix) array which require significantly less memory than suffix trees [26] but which can still be constructed in linear time [27, 28, 29].

One of the major difficulties in comprehensive repeat analysis lies in the visualisation of repeated structures. Additionally, repeat visualisation, as it relates to word frequency, may also be of interest in the linguistic analysis of genomic sequences [30] and, as shall be seen, parallels between human language and genomic repeat visualisation can be drawn.

Genomes are typically too large to be efficiently visualised as a string of symbols or to be represented as a line. When the additional problem of identifying and categorising repeats that differ widely in length, position and spatial relationship is added the problem becomes increasingly challenging.

Some attempts at sequence visualisation have been made such as the visualisation of tandem repeats using colour-coding [14], the side-by-side comparison of simple repeats [20] and visual linking of maximal repeats between two strands [2]. Of these, GenAlyzer [31] and its predecessor Reputer [2] are the only tools that visualise all maximal repeats within a sequence. The GenAlyzer visualisation consists of two horizontal lines, both of which may represent the same sequence, repeats are shown as lines connecting the positions of repeat on the two sequences. This provides a natural way of viewing all maximal repeats and their distribution. However the visualisation has the potential to become saturated for long, or highly repetitive sequences. (see [2] figure 7).

This paper presents an efficient algorithm for collecting all exact repeats within a genome. This data is presented as a colour plot which, through visual inspection, exposes some of the many complex repeat types in these sequences. This is the first time all repeats have been visualised in a single plot. The work highlights the extent to which repeats occur within genomic sequences and gives some indication of the important role repeats play. From this it is clear that large scale repeat analysis remains an important, and largely open, problem.

2. Results

In order to create the visualisation we consider all possible substrings of length k in a sequence. The number of repetitions r of each substring is counted (i.e. for substrings that occur once $r = 0$, for substrings that occur twice $r = 1$, etc.). The repeat score function $C(k, r)$ is the number of substrings that repeat r-times, for a given substring length k. For example if there are 30 different sequences of 20 nucleotides that each occur 15 times in a specific genome then $C(20, 15)$=30.

Figure 1a shows an example of the repeat score function $C(k, r)$ for several values of r calculated for the whole human genome (build 35.1 [4]). The repeat score function for $r = 0$, i.e. no repeats, shows the amount of unique sequences within the genome as a function of substring length k and was previously used in the analysis of sequence reassembly [32].

For $r > 0$ the repeat score falls by several orders of magnitude as shown in figure 1a and is the main reason for employing a log scale.

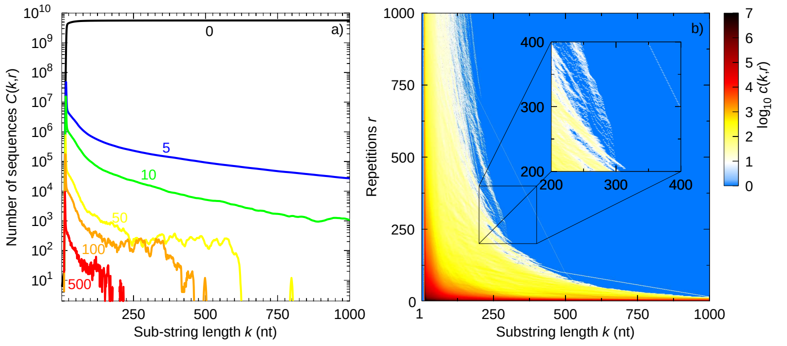

Figure 1. Repeat score function $C(k,r)$ for the whole human genome (build 35.1 [4]) (a) as a function of substring length k and selected values of r and (b) a logarithmic colour plot of the repeat score function $C(k,r)$, where the x and y axis indicate the substring length k and number of repeats r respectively, the colour of the point at position (k,r) indicates the number of differently composed substrings that repeat r times. The inset highlights an example of the complex structure present in this plot.

Similarities in repeat score function $C(k,r)$ can be seen for specific repeat values r: For instance for values of k between 250 and 400 the repeat score function of the human genome produces similar values for $C(k,r)$ where r is equal to 50 or 100. However outside this range there are strong differences. It is therefore interesting to plot the repeat score continuously as a function of r. Such a repeat score plot is shown in figure 1b. The plot shows the repeat score function $C(k,r)$ as a function of both the substring length k and the number of repeats, r, for the whole human genome, where colour represents $\log_{10} C(k,r)$.

It is clear from figure 1b that representing the repeat score function $C(k,r)$ in this way allows the rapid visualisation of the complex repeat structure of a sequence. This can be seen more clearly for the human genome when specific features of the plot are expanded, figure 1b. Figure 1 also illustrates the capability of the underlying algorithm to analyse large genomes. Since we consider both strands of DNA this analysis was performed on a string of 6.2 billion bases. Clearly, algorithms with (close to) linear time complexity are necessary for studying such genomes.

To further understand and identify the structure present in this visualisation a series of artifical sequences were constructed which provide some basis for the interpretation of the repeat score plots. Figure 2a shows the repeat score plot for an artifically generated sequence containing three types of very simple periodic repeat: a sequence composed only of a mononucleotide A; a triplet repeat ATG; and a quadruplet repeat $ATGC$ each repeated 20 times. Such simple repeats always appear as straight lines in the repeat score plot. However, the slope of this line depends both on the size of the repeated section, and on how many times they are continuously repeated. Therefore its interpretation from the visualisation alone is not trivial. However the sequence data from which a feature is composed may be extracted for further investigation.

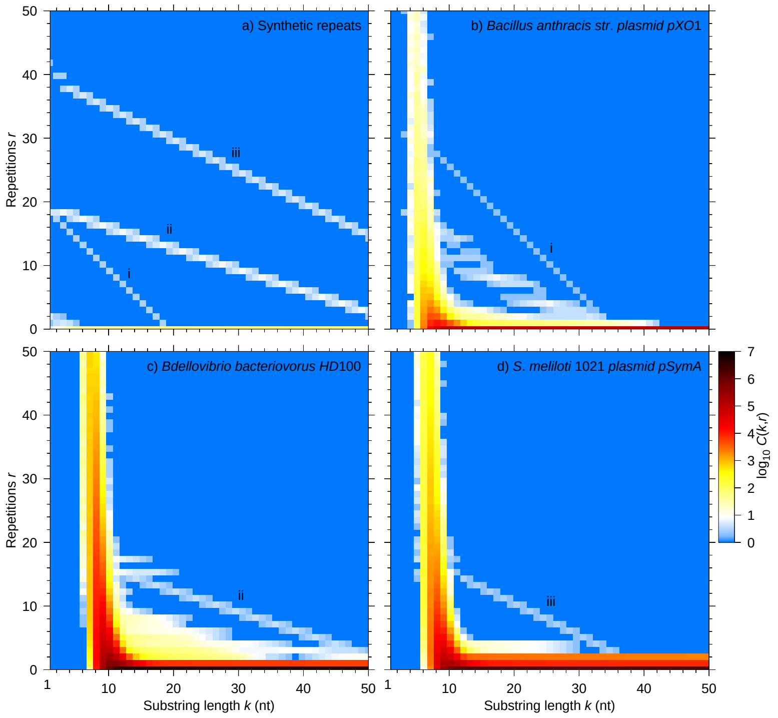

Figure 2. Logarithmic colour plot of the repeat score function $C(k, r)$ for (a) A constructed sequence composed of (i) A repeated 20 times, (ii) the triple ATG repeated 20 times and (iii) the quadruple $ATGC$ repeated 20 times. (b) *Bacillus anthracis*, size 95 kb [GenBank:NC_007323], (c) *Bdellovibrio bacteriovorus* HD100, size 3.8 Mb [GenBank:NC_005363], and (d) *Sinorhizobium meliloti* 1021 plasmid pSymB, size 1.7 Mb [GenBank:NC_003037]. Parts (b), (c) and (d) contain identified simple repeat structures marked as (i), (ii) and (iii) as in part (a).

Randomly generated sequences contain few repeats, and the repeat score plot of such sequences, as shown in figure 3a, typically presents strong clustering around small substring lengths k and repetitions r. The repeat score decreases rapidly for moderately long substrings. For

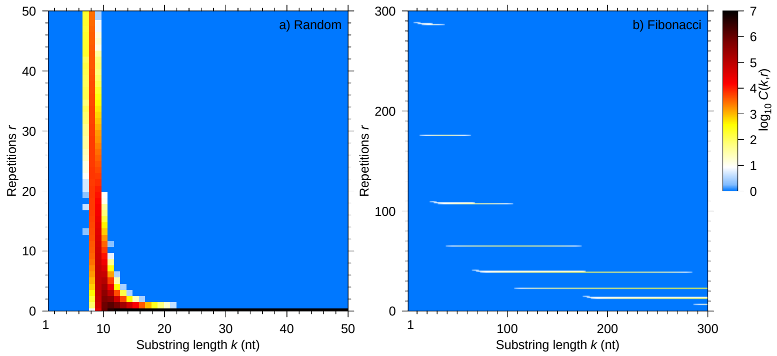

Figure 3. Logarithmic colour plot of the repeat score function $C(k, r)$ for (a) a random genomic sequence of size 4.7 Mb and (b) a Fibonacci sequence a_{15}, with $a_1 = AT$ and $b_1 = GC$, generated from Eq. 1.

the sequence shown in figure 3a (length 4.7 Mb) the base composition was biased as in [33]. When random sequences of increasing length are analysed the plot retains the same general shape, while the ridges shift to larger lengths and greater numbers of repetitions. The random sequence therefore provides an important baseline for comparison with other sequences and represents an absence of repeat structure. However, absence of repeat structure should not be confused with absence of functional structures containing information such as those that code for proteins. While the prokaryotic and eukaryotic genomes analysed showed strong deviation from the randomly generated sequences (see below), the analysis of small viral genomes yields a repeat score plot similar to that of a random sequence of the same size (data available from our website: `http://4g.soton.ac.uk`). This is consistent with a lack of redundancy in the genomes of these highly efficient organisms.

Another interesting example, which highlights the potential of the repeat score plot, are quasi-periodic structures, i.e., sequences with a high degree of structure and periodicity but yet not completely periodic [34]. A well known example is the Fibonacci sequence, created by taking two seed sequences, in this case: $a_1 = AT$ and $b_1 = GC$, and applying the inflation rule

$$a_{i+1} = a_i + b_i, \quad b_{i+1} = a_i. \tag{1}$$

For instance a sequence a_5 would be

$$a_5 = ATGCATATGCATGCAT$$

In figure 3b the repeat score for a Fibonacci sequence a_{15} (3195 nt) is shown. Although the Fibonacci sequence has a high degree of repetition its repeat score function shows a distinctive difference from the sloped lines of simple repeats, figure 2a. Instead horizontal lines are observed which become broader and more closely spaced for lower repetitions r. These features are characteristic of all the Fibonacci sequences we have analysed, regardless of size or seed sequences used (data not shown). This indicates that the repeat score plot allows the visualisation of repeats based upon the repeat structure, not the size or base composition of this particular sequence. Such structures might be expected to be generated by serial duplication of a specific region. In a visual inspection of over 900 real sequences we were unable to find any displaying this characteristic pattern. To date, no Fibonacci sequences have been found. Our study suggests that if they do occur they are quite rare.

As a general rule, highly repetitive structures appear in the repeat score plot at large substring lengths k and many repetitions r, while non-repeated structures show as intense clusters for small values of k and r. The logarithmic plot employed in our analysis ensures that only significant repeats are highlighted, although there is no restriction in using a linear scale when less frequent repeats are to be detected.

2.1 Visualisation of genomic sequences

From our analysis of artifical sequences it becomes clear that the repeat score function visualised as a function of substring length and number of repetitions provides a visual “signature” which is helpful in identifying repeated elements in genomic sequences. Figure 2b,c,d shows examples of simple repeat structures found in real genome sequences. These were identified by visual comparison with artifically generated repeat score plots, such as those shown in figure 2a. Subsequently, the sequence data related to these repeat elements were extracted and confirmed to be of a similar type to the synthetic repeats. That is to say, the structure labelled i, in figures 2a and 2b was a mononucleotide repeat composed of the nucleotide A. The structure labelled ii was in both cases a triplet repeat (ATG in the artificial sequence and GAA in figure 2b), and structure iii was composed of a quadruplet repeat ($ATGC$ in the artificial sequence and $AGAG$, in figure 2d, showing that two and four nucleotide repeat elements will exhibit a similar structure). Note that the repeats were identified independently of the size of the genome, which in our examples ranged from 95 kb (figure 2a) to 3.8 Mb (figure 2b).

Apart from detecting specific repeat patterns, the repeat score can also be used for comparative analysis between different regions of a genome. For instance, the coding and non-coding sections of a genome can be visualised separately and then compared, even if the two sections are very different in size.

In figure 4a the repeat score plot of the coding section of the well

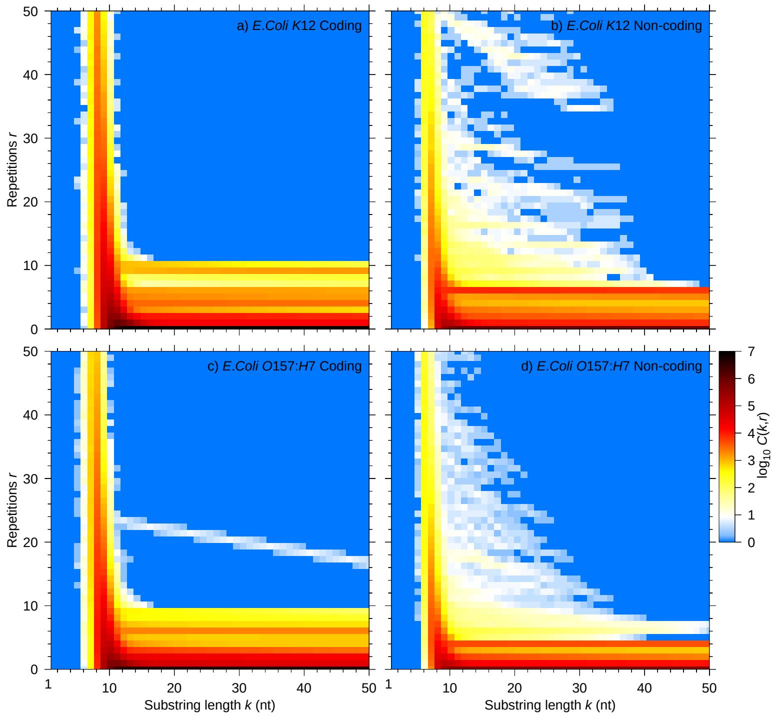

Figure 4. Logarithmic colour plot of the repeat score function $C(k,r)$ for (a) the coding and (b) non-coding portion of non-pathogenic *E. coli* K12, size 4.6 Mb [GenBank:NC_000913], and (c) the coding and (d) non-coding portion of the pathogenic *E. coli* O157:H7, size 5.5 Mb [GenBank:NC_002695].

annotated *E. coli* K12 genome [GenBank:NC_000913] is shown. First a strong clustering around substring length $k = 10$ nt and repetitions $r = 10$ is noted, similar to the random sequence (see figure 3a), and then the broad horizontal lines, at low values of r, which indicate large exact repeats. These are sequences that occur up to four times and are largely made up of ancient proteins involved in protein translation, tRNA metabolism and proteins involved in transposition or related to prophages. Another identified feature is the peak at $r = 6$ repetitions which arises from the seven copies of genes coding for components of the ribosome [35]. The repeat score plot of the non-coding part is very different, with a highly repetitive cloud-like pattern clearly visible in figure 4b. The fact that the non-coding sequence accounts for only 15% of the genome was of no importance for the detection of these patterns, which are also clearly visible in the analysis of the complete genome (data not shown). Many of these repeats are intergenic repetitive sequences such as Intergenic Repeat Elements (IRUs) and Enterobacterial Repetitive Intergenic Consensus sequences (ERICs) [13]. ERICs are related to the REP family of repeats for which multiple functions have been proposed including transcription termination, mRNA stability and chromosomal domain organisation *in vivo* [36]. The exclusive presence of highly repeated short sequences within non-coding DNA appears to be a characteristic of a large number of the sequences we have analysed (data available from our website: `http://4g.soton.ac.uk`).

Since size poses no significant constraint for the repeat score plot the comparison of different genomes can easily be performed. Figures 4c and 4d show the coding and non-coding regions, respectively, of a pathogenic strain of *E. coli* [GenBank:NC_002695]. Comparing this to the non-pathogenic *E. coli* of figures 4a,b is straightforward, despite a difference in size of almost 1 Mb. While the coding and non-coding regions of these two genomes are largely similar, the striking diagonal feature in figure 4c exists only in the coding region of the pathogenic strain. The sequences composing this feature were identified as part of a pathogenicity island [37], structures found within the genomes of microbes and pathogens that are associated with pathogenicity. These regions are often flanked by direct repeats, generated by the integration of the pathogenicity island into the host genome via recombination [38]. Similar features in the genomes of many pathogenic organisms such as *Mycoplasma pneumoniae M129* [Genbank:NC_000912.1] and *Haemophilus influenzae Rd KW20* [Genbank:NC_000907.1] were observed. Repeat score plots for these and more than 500 other sequences can be accessed from our website (`http://4g.soton.ac.uk`).

3. Discussion

A visualisation technique that can rapidly identify all repeated sequences of any length in genomes up to and including the 6.1 billion bases (including the forward and reverse strand) of the full human genome has been developed. The plots expose the complex repeat structure that exists in genomic sequences. They provide a visualisation that is largely independent of genome size and can identify repeated sequences independently of their position and spatial relationship. Unlike many techniques for locating repeats, the repeat score plot can be used to locate repeats of all sizes, and those of unknown type and structure. The plots provide a signature of the repeat structure of a sequence as well as providing a straightforward means for comparison within and between sequences. While the visualisation is largely independent of genome size, normalising the visualisation with respect to genome size would be a useful extension of this work. One possible solution to this problem would be to plot the degree to which repeats

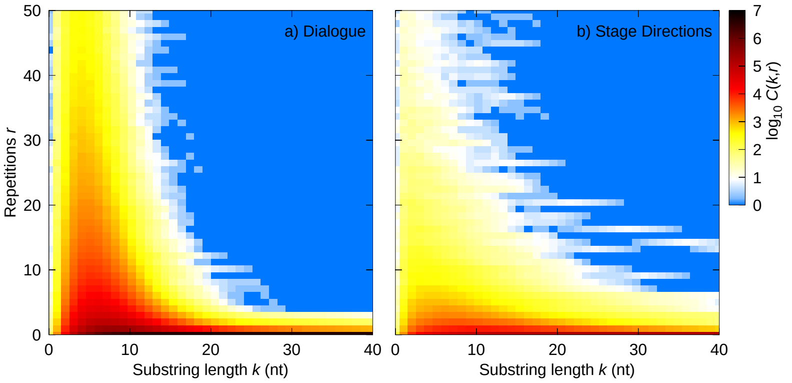

Figure 5. Logarithmic contour plot of the repeat score function $C(k, r)$ for the a) dialogue and b) stage directions of the plays of Shakespeare [39]. The dialogue not only constitutes the largest part of the text; it is also the most creative part, for which a relatively low level of repetition would be expected. On the other hand, stage directions are highly repetitive. 'Act 1 Scene 1' appears in every one of the 37 plays. The figure illustrates this distinction. An analogy can be drawn between the dialogue and the coding information in the genome sequences and the stage directions and non-coding DNA. Indeed if there were many more plays by Shakespeare (i.e. 'Act 1 Scene 1' occurred 100 or 200 times rather than 37) then the similarity between repeat frequency in the non-coding DNA and the stage directions would be even stronger.

deviate from a random distribution. Constructing an effective random model, however, is a non-trivial problem which we hope to investigate in the future.

Using this technique significant differences between the repeat structure of the coding and non-coding regions of prokaryotic and eukaryotic genomes have been identified. As have striking features associated with pathogenicity in bacterial genomes.

The most important contribution of this work is to describe the extent to which repetitions of functional importance occur in genomic sequences. We have shown that these are not merely simple repeats but in many cases have a complex structure which warrants further investigation.

The technique may also prove useful as an analysis in its own right. It may, for example, be useful in annotation problems, such as the analysis of newly sequenced genomes. Where structure is unknown, repeated elements can be quickly and easily identified using this technique.

The use of these visualisations is obviously not limited to genetic sequences but can be applied to any string. Therefore it will have application to the analysis of language like features within genomic sequences [30] as well as potential applications in the analysis of language and data compression. An example of its application to written language which shows parallels with genomic analysis, is shown in figure 5.

The plot has potential as an analytical tool and the development of normalised plots and objective comparison is clearly the most important next step. However a particular strength is its ability to rapidly present a visual representation of the repeat structure of a very large string. The importance of fully utilising the power of the human visual system to recognise patterns has been discussed [40] and this representation approach provides a novel method of exploiting this.

Appendix

A. Algorithm

An all against all comparison approach to constructing the repeat score plot would require $\mathrm{O}(kn^2)$ comparisons. Here an optimal algorithm is described.

Consider a string $S[0:n]$ with elements indexed from 0 to n where the last element is marked with a special end marker $S[n] = \$$ which does not occur at any other position. The suffix array [24] $sa[0:n]$ of all suffixes and the array $lcp[1:n]$ giving the length of the longest common prefix between $sa[k]$ and $sa[k-1]$ can efficiently be computed in linear time [29, 27, 28].

An example string, 'banana$', shall be used to illustrate the operation of the algorithm. This has suffixes banana$, anana$, ..., a$, $. The suffix array and lcp array for this example are given in table 1.

A relation between suffixes $R_k(s1, s2)$ is defined which is true if $s1$ and $s2$ share the same prefix of length k. The equivalence relation $R_k(s1, s2)$ defines a partitioning of the suffixes into equivalence classes. Equivalence classes can be labelled by $[prefix]$ where $prefix$ is the

Index	Position	Suffix	lcp
0	1	anana$	
1	3	ana$	3
2	5	a$	1
3	0	banana$	0
4	2	nana$	0
5	4	na$	2
6	6	$	0

Table 1. Suffix and LCP array of the string `banana$`

Index	Suffix	lcp	$k = 1$	$k = 2$	$k = 3$	$k = 4$
0	anana$		[a]	[an]	[ana]	[anan]
1	ana$	3	[a]	[an]	[ana]	[ana$]
2	a$	1	[a]	[a$]	[a$.]	[a$..]
3	banana$	0	[b]	[ba]	[ban]	[bana]
4	nana$	0	[n]	[na]	[nan]	[nana]
5	na$	2	[n]	[na]	[na$]	[na$.]
6	$	0	[$]	[$.]	[$..]	[$...]

Table 2. Equivalences classes for substrings of `banana$`

prefix which all strings share. For our example the equivalence classes are shown in table 2.

When the suffix is shorter than k an arbitrary symbol '.' is added. The equivalence classes are denoted by

$$\{0, 1, \ldots, n-1\}/R_k = \{\mathcal{P}_k^i | i = 1, n_k\} \tag{A.1}$$

where n_k is the number of equivalent classes for substrings of size k and in our example

$$\begin{aligned}
\mathcal{P}_1^1 &= [\mathsf{a}] = \{\mathsf{anana\$}, \mathsf{ana\$}, \mathsf{a\$}\} \\
\mathcal{P}_1^2 &= [\mathsf{b}] = \{\mathsf{banana\$}\} \\
\mathcal{P}_1^3 &= [\mathsf{n}] = \{\mathsf{nana\$}, \mathsf{na\$}\} \\
\mathcal{P}_1^4 &= [\mathsf{\$}] = \{\mathsf{\$}\} \\
\mathcal{P}_2^1 &= [\mathsf{an}] = \{\mathsf{anana\$}, \mathsf{ana\$}, \mathsf{a\$}\} \\
\mathcal{P}_2^2 &= [\mathsf{a\$}] = \{\mathsf{a\$}\} \\
\mathcal{P}_2^3 &= [\mathsf{ba}] = \{\mathsf{banana\$}\} \\
\mathcal{P}_2^4 &= [\mathsf{na}] = \{\mathsf{nana\$}, \mathsf{na\$}\} \\
\mathcal{P}_2^5 &= [\mathsf{\$.}] = \{\mathsf{\$}\}
\end{aligned}$$

etc. The number of substrings of length k that repeats r times is equal to

$$C(k, r) = \sum_{i=1}^{n_k} \llbracket |\mathcal{P}_k^i| = r \rrbracket \tag{A.2}$$

where $\llbracket \textit{predicate} \rrbracket$ is equal to one if *predicate* is true and equal to zero otherwise.

Let v_{max} be the largest value in the lcp array. It is noticed that for $k > v_{max}$ all members suffixes are unique (by the definition of v_{max}). Therefore for $k > v_{max}$, $C(k, 0) = n + 1 - k$ and $C(k, r) = 0$ for $r > 0$. The equivalence classes $\mathcal{P}_k^1$ can now be computed iteratively starting from $k = v_{max}$ and decrementing until $k = 1$ is reached. This is performed efficiently by sorting the lcp array into a list of sets indexed by the value of the lcp array, where the elements in the set are the indexes of the lcp array. That is, $i \in \mathcal{H}[v]$ if $lcp[i] = v$. In our example,

$$\begin{aligned}\mathcal{H}[0] &= \{3, 4, 6, 7\}\\ \mathcal{H}[1] &= \{2\}\\ \mathcal{H}[2] &= \{5\}\\ \mathcal{H}[3] &= \{1\}.\end{aligned}$$

The algorithm uses a *disjoint set* data structure [41] to efficiently compute union operations. The *disjoint set* data structure is augmented with an auxiliary array which maintains the sizes of sets. $C[v_{max} + 1, 0]$ is first initialised to be $|lcp|$, this can be seen to be so because for substrings greater than v_{max} all substrings are unique (by definition of v_{max}).

$\mathcal{H}$ is iterated over from $v = v_{max}$ to $v = 1$. For each value of v, the sizes of all sets in $\mathcal{H}[w]$ where $w \geq v$ are obtained, and stored in C such that $C[v][x]$ contains the number of sets of size $x + 1$ (1 is added to maintain our original definition of $C(k, 0)$).

To do this $C[v]$ is initialised with $C[v + 1]$. Union operations are then performed on $\mathcal{H}$, first subtracting sizes of the sets to be unioned, then adding the size of the final unioned set. The algorithm can be summarised in the following pseudo code:

```
Input:  Array of sets $\mathcal{H}[1 : v_{max}]$
        Extended lcp array $lcp[0 : n]$
Output: Count of repeats $C[1 : n, 0 : n]$

Initialise DisjSets;
$C[v_{max} + 1, 0]$ $\leftarrow |lcp|$

for $v$ $\leftarrow v_{max}$ to 1
    $C[v, 0..n]$ $\leftarrow C[v + 1, 0..n]$
    forall $i \in \mathcal{H}[v]$
        $j$ $\leftarrow$ DisjSets.find($i$)
        $k$ $\leftarrow$ DisjSets.find($i - 1$)

        $C[v, |j| - 1]$ $\leftarrow C[v, |j| - 1] - 1$
        $C[v, |k| - 1]$ $\leftarrow C[v, |k| - 1] - 1$

        $l$ $\leftarrow$ DisjSets.union($j$,$k$)

        $C[v, |l| - 1]$ $\leftarrow C[v, |l| - 1] + 1$
    endfor
endfor
```

The algorithm can be modified to compensate for the unique end marker and substrings overlapping the end of the sequence, simply by subtracting k, $C'(k,0) = C(k,0) - k$. Sequence breaks may be compensated for similarly.

It is clear that the algorithm operates in linear time, a single operation for every entry in the array $\mathcal{H}$. The complexity of a given implementation will therefore depend on the information you wish to extract, which in the case of the repeat score plot, will be its area.

B. Example Implementation

In this section an example C++ implementation of the algorithm previously described in provided. When combined with a suffix and LCP construction algorithm, this forms a complete implementation of the algorithm described. The function `repeatscore` takes the lcp array of a string as its input and returns a newly constructed 2 dimensional vector containing the repeatscore matrix of the input sequence indexed by substring length and number of repeats. A complete implementation is available on request from the corresponding author.

```
#include <vector>
#include <iostream>

using namespace std;
typedef vector<vector<int> > vec2d;

class DisjSets {
public:
  vector<int> s, s_sizes;
  DisjSets(int l) : s(l,-1) , s_sizes(l,1) {}

  int find(int x) {
    if(s[x] < 0) return x;
    return s[x] = find(s[x]);
  }

  int unionsets(int r1,int r2) {
    if(s[r2] < s[r1]) {int sw=r1; r1=r2; r2=sw;}

    if(s[r1]==s[r2]) s[r2]--;
    s_sizes[r1]=s_sizes[r1]+s_sizes[r2];
    return s[r2]=r1;
  }

  int set_size(int x) { return s_sizes[x]; }
};

int repeatscore(vector<int> &lcp,vec2d **C_ptr) {
  int n=lcp.size();
  DisjSets p(n);

  int v_max=0;
  for(int i=0;i<n;i++) // find v_max
   if(lcp[i] > v_max) v_max=lcp[i];
  vec2d height(v_max+1,vector<int>(0));
  for(int i=1;i<n;i++) // create height
    height[lcp[i]].push_back(i);

  *C_ptr = new vec2d(v_max+2,vector<int>(n,0));
  vec2d &C = **C_ptr;

  C[v_max+1][0]=n;

  for(int k=v_max;k>0;k--) {
    for(int cp=0;cp<n;cp++) C[k][cp]=C[k+1][cp];

    vector<int>::iterator i = height[k].begin();
    for(;i != height[k].end();i++) {
      int s = p.find((*i));
      int t = p.find((*i)-1);

      C[k][p.set_size(s)-1]--; // remove old
      C[k][p.set_size(t)-1]--;

      int l = p.unionsets(s,t);

      C[k][p.set_size(l)-1]++; // add new set
    }
  }
}
```

C. Acknowledgements

This work was supported by Research Councils UK through the Basic Technology Programme. Processing time was made available by ISS at the University of Southampton who we thank for their support.

References

[1] Alex van Belkum, Stewart Scherer, Loek van Alphen, and Henri Verbrugh. Short-Sequence DNA Repeats in Prokaryotic Genomes. *Microbiol. Mol. Biol. Rev.*, 62(2):275–293, 1998.

[2] Stefan Kurtz, Jomuna V. Choudhuri, Enno Ohlebusch, Chris Schleiermacher, Jens Stoye, and Robert Giegerich. REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Research*, 29(22):4633–4642, 2001.

[3] Guillaume Achaz, Eric Coissac, Pierre Netter, and Eduardo P. C. Rocha. Associations Between Inverted Repeats and the Structural Evolution of Bacterial Genomes. *Genetics*, 164(4):1279–1289, 2003.

[4] E Lander et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):745–964, 2001.

[5] Eduardo P. C. Rocha, Antoine Danchin, and Alain Viari. Analysis of long repeats in bacterial genomes reveals alternative evolutionary

mechanisms in *Bacillus subtilis* and other competent prokaryotes. *Mol. Biol. Evol.*, 16(9):12191230, 1999.

[6] Anthony M. Paiva and Richard D. Sheardy. Influence of sequence context and length on the structure and stability of triplet repeat DNA oligomers. *Biochem.*, 43:14218–14227, 2004.

[7] Masataka Tsuge, Ryuji Hamamoto, Fabio Pittella Silva, Yozo Ohnishi, Kazuaki Chayama, Naoyuki Kamatani, Yoichi Furukawa, and Yusuke Nakamura. A variable number of tandem repeats polymorphism in an E2F-1 binding element in the 5' flanking region of SMYD3 is a risk factor for human cancers. *Nat. Genet.*, 37:1104 – 1107, 2005.

[8] Subbaya Subramanian, Rakesh Mishra, and Lalji Singh. Genome-wide analysis of microsatellite repeats in humans: their abundance and density in specific genomic regions. *Genome Biol.*, 4(2):R13, 2003.

[9] Jeff Bizzaro and Kenneth Marx. Poly: a quantitative analysis tool for simple sequence repeat (SSR) tracts in DNA. *BMC Bioinformatics*, 4(1):22, 2003.

[10] Akito Taneda. Adplot: detection and visualization of repetitive patterns in complete genomes. *Bioinformatics*, 20(5):701708, 2004.

[11] Jeff Reneker and Chi-Ren Shyu. Refined repetitive sequence searches utilizing a fast hash function and cross species information retrievals. *BMC Bioinformatics*, 6(1):111, 2005.

[12] Alkes L. Price, Neil C. Jones, and Pavel A. Pevzner. De novo identification of repeat families in large genomes. *Bioinformatics*, 21(suppl_1):i351–358, 2005.

[13] S. Bachellier, J-M Clément, and M Hofnung. Short palindrome repetitive DNA elements in enterobacteria: a survey. *Res. Microbiol.*, 150:627–639, 1999.

[14] Tetsuhiko Yoshida, Nobuaki Obata1, and Kenji Oosawa. Color-coding reveals tandem repeats in the *Escherichia coli* genome. *J. Mol. Biol.*, 298:343–349, 2000.

[15] G Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucl. Acids. Res.*, 27(2):573–580, 1999.

[16] Adalberto T. Castelo, Wellington Martins, and Guang R. Gao. TROLL — Tandem Repeat Occurrence Locator . *Bioinformatics*, 18(4):634–636, 2002.

[17] Roman Kolpakov, Ghizlane Bana, and Gregory Kucherov. mreps: efficient and flexible detection of tandem repeats in DNA. *Nucl. Acids. Res.*, 31(13):36723678, 2003.

[18] D. Sharma, B. Issac, G. P. S. Raghava, and R. Ramaswamy. Spectral repeat finder (SRF): identification of repetitive sequences using Fourier transformation. *Bioinformatics*, 20(9):1405–1412, 2004.

[19] Alex van Belkuma, Willem van Leeuwena, Stewart Schererb, and Henri Verbrugha. Occurrence and structure-function relationship of pentameric short sequence repeats in microbial genomes. *Res. Microbiol.*, 150:617–626, 1999.

[20] Jeong-Hyeon Choi and Hwan-Gue Cho. Analysis of Common k-mers for Whole Genome Sequences Using SSB-Tree. *Genome Informatics*, 13:30–41, 2002.

[21] Natalia Volfovsky, Brian Haas, and Steven Salzberg. A clustering method for repeat analysis in DNA sequences. *Genome Biol.*, 2(8):research0027.1–research0027.11, 2001.

[22] Mohamed Ibrahim Abouelhoda, Enno Ohlebusch, and Stefan Kurtz. Optimal exact string matching based on suffix arrays. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, pages 31–43. Springer-Verlag, 2002.

[23] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. The enhanced suffix array and its applications to genome analysis. In *WABI '02: Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, pages 449–463, London, UK, 2002. Springer-Verlag.

[24] Udi Manber and Gene Myers. Suffix arrays: A new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948, 1993.

[25] G Manzini. Two space saving tricks for linear time LCP computation. Technical Report 124, Universita del Piemonte, Dipartimento di Informatica, 2004.

[26] P. Weiner. Linear pattern matching algorithms. *IEE 14th Ann. Symp. on Switching and Automata Theory*, 1:1–11, 1973.

[27] Pang Ko and S Aluru. Space-efficient linear time construction of suffix arrays. In *Accepted to Symp. Combinatorial Pattern Matching*, 2003.

[28] Dong Kyue Kim, Jeong Seop Sim, Heejin Park, and Kunsoo Park. Linear-Time Construction of Suffix Arrays. In *Proc. 14th Annual Symposium on Combinatorial Pattern Matching*, 2003.

[29] Juha Kärkkäinen and P Sanders. Simple linear work suffix array construction. In *Proc. Int. Colloq. Automata Languages and Programming*, pages 943–955. Springer, 2003.

[30] David B. Searls. The language of genes. *Nature*, 420:211–217, 2002.

[31] Jomuna V. Choudhuri, Chris Schleiermacher, Stefan Kurtz, and Robert Giegerich. GenAlyzer: interactive visualization of sequence similarities between entire genomes. *Bioinformatics*, 20(12):1964–1965, 2004.

[32] Nava Whiteford, Niall Haslam, Gerald Weber, Adam Prügel-Bennett, Jonathan W. Essex, Peter L. Roach, Mark Bradley, and Cameron Neylon. An analysis of the feasibility of short read sequencing. *Nucl. Acids. Res.*, 33(19):e171–, 2005.

[33] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*, page 50. Cambridge Univ. Press, 2000.

[34] Sara Cuenda and Angel Sánchez. Nonlinear excitations in DNA: Aperiodic models versus actual genome sequences. *Phys. Rev. E*, 70:051903, 2004.

[35] Frederick R. Blattner, III Plunkett, Guy, Craig A. Bloch, Nicole T. Perna, Valerie Burland, Monica Riley, Julio Collado-Vides, Jeremy D. Glasner, Christopher K. Rode, George F. Mayhew, Jason Gregor, Nelson Wayne Davis, Heather A. Kirkpatrick, Michael A. Goeden, Debra J. Rose, Bob Mau, and Ying Shao. The complete genome sequence of *Escherichia coli* K-12. *Science*, 277(5331):1453–1462, 1997.

[36] James Versalovic, Thearith Koeuth, and R. Lupski. Distribution of repetitive DNA sequences in eubacteria and application to finerpriting of bacterial enomes. *Nucl. Acids Res.*, 19(24):6823–6831, 1991.

[37] Jorg Hacker, Larisa Bender, Manfred Ott, Jochen Wingender, Bjorn Lund, Reinhard Marre, and Werner Goebel. Deletions of chromosomal regions coding for fimbriae and hemolysins occur in vitro and in vivo in various extra intestinal Escherichia coli isolates. *Microb. Pathog.*, 8:213–225, 1990.

[38] Jorg Hacker and James B. Kaper. Pathogenicity islands and the evolution of microbes. *Annual Review of Microbiology*, 54(1):641–679, 2000.

[39] Jon Bosak. The plays of Shakespeare, 1999. http://www.ibiblio.org/bosak.

[40] Stephen Wolfram. *A New Kind of Science*, page 548. Wolfram Media, 2002.

[41] Benrard A. Galler and Michael J. Fisher. An improved equivalence algorithm. *Commun. ACM*, 7(5):301–303, 1964.