Bioinformatics, 35(6), 2019, 914–922
doi: 10.1093/bioinformatics/bty747
Advance Access Publication Date: 28 August 2018
Original Paper

OXFORD

Genome analysis

Dot2dot: accurate whole-genome tandem repeats discovery

Loredana M. Genovese[1,†], Marco M. Mosca[3,†], Marco Pellegrini[1,2] and Filippo Geraci [1,*]

[1]Institute for Informatics and Telematics, CNR, Pisa 56124, Italy, [2]Laboratory of Integrative Systems Medicine (LISM), Institute of Informatics and Telematics and Institute of Clinical Physiology, 56124 Pisa, Italy and [3]Department of Computer Science, University of Liverpool, L69 3BX Liverpool, UK

*To whom correspondence should be addressed.
†The authors wish it to be known that these authors contributed equally.
Associate Editor: John Hancock

Received on March 30, 2018; revised on August 3, 2018; editorial decision on August 23, 2018; accepted on August 24, 2018

Abstract

Motivation: Large-scale sequencing projects have confirmed the hypothesis that eukaryotic DNA is rich in repetitions whose functional role needs to be elucidated. In particular, *tandem repeats* (TRs) (i.e. short, almost identical sequences that lie adjacent to each other) have been associated to many cellular processes and, indeed, are also involved in several genetic disorders. The need of comprehensive lists of TRs for association studies and the absence of a computational model able to capture their variability have revived research on discovery algorithms.
Results: Building upon the idea that sequence similarities can be easily displayed using graphical methods, we formalized the structure that TRs induce in dot-plot matrices where a sequence is compared with itself. Leveraging on the observation that a compact representation of these matrices can be built and searched in linear time, we developed *Dot2dot*: an accurate algorithm fast enough to be suitable for whole-genome discovery of TRs. Experiments on five manually curated collections of TRs have shown that *Dot2dot* is more accurate than other established methods, and completes the analysis of the biggest known reference genome in about one day on a standard PC.
Availability and implementation: Source code and datasets are freely available upon paper acceptance at the URL: https://github.com/Gege7177/Dot2dot.
Contact: filippo.geraci@iit.cnr.it
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

A tandem repeat (TR) consists in a certain number of copies of a (typically small) motif sequence that occur adjacent to each other. More realistic definitions admit a certain degree of heterogeneity among copies of the motif sequence as well as tiny insertions or deletions. The abundance of these structures in eukaryotic DNAs has been observed since the first sequencing data became available in the early 90s (Bacolla *et al.*, 2008). Although TRs functional role is not completely understood yet, their distribution in eukaryotic genomes suggests involvement in several cellular processes including gene expression (Tóth *et al.*, 2000). Besides confirming this thesis, the steady growth of the number of genetic disorders related to the expansion of TRs has kindled the hope of associating TRs polymorphism with the etiology of those genetic diseases that are still unexplained. This trend led the bioinformatics community to focus on research projects aimed at a large-scale analysis of repetitions. Unfortunately, validating a new relevant TR can be as difficult as finding a needle in the haystack and the success of these projects heavily depends on the sensitivity of the searching algorithms. The difficulty of capturing the variability of satellites and microsatellites into a single comprehensive computational model has encouraged researchers to design new methods for large-scale TR discovery.

Nevertheless, an agreed computational model for non-naive biologically relevant TRs is still far away.

According to different searching strategies and definitions of TRs, several algorithm families have been proposed. Dictionary-based methods (Castelo *et al.*, 2002; Delgrange and Rivals, 2004; Karaca *et al.*, 2005; Parisi *et al.*, 2003; Pokrzywa and Polanski, 2010) leverage on a pre-determined set of seeds that is searched along the input sequence and subsequently expanded. This class of algorithms is particularly efficient in those situations where the user is interested in a relatively small class of repeats. Exhaustive search of TRs, instead, is unaffordable because of the exponential growth of the dictionary size as a function of the allowed motif length. The dual approach is that of *ab-initio* methods (Benson, 1999; Boeva *et al.*, 2006; Girgis and Sheetlin, 2013; Kofler *et al.*, 2007; Kolpakov *et al.*, 2003; Krishnan and Tang, 2004; Kurtz *et al.*, 2001; Pellegrini *et al.*, 2010; Sokol *et al.*, 2007; Wexler *et al.*, 2005; Wirawan *et al.*, 2010; Zhou *et al.*, 2009) that do not require any pre-existing knowledge about the input sequence. Although more complex, this class of algorithms is the most studied and used in practice.

The presence of variations (single nucleotide polymorphisms, insertions and deletions) in the genomes has directed many researchers to develop algorithms in which the distance between units of a motif is based on the Needleman–Wunsch sequence alignment algorithm described in Needleman and Wunsch (1970). The high quadratic cost of this procedure, however, has convinced researchers to shift to algorithms that work in Hamming distance. Hybrid approaches use Hamming distance instead of sequence alignment but allow insertions between consecutive copies of the motif.

Output filtering is a desirable but not mandatory feature of TR searching methods. Its main advantage is the elimination/reduction of redundancy often caused by software artifacts. Aggressive filtering, however, can cause the removal of relevant results and, as a consequence, the reduction of the algorithm accuracy.

Two distinguishing features have recently gained importance: the ability of managing multi-sequence files coming from NGS sequencing and the possibility of scanning entire assembled genomes. Both these features can involve a potentially huge amount of data and thus require fast algorithms that avoid computationally expensive operations without sacrificing output quality.

In this paper, we present *Dot2dot*, a novel algorithm for TR discovery. Our method borrows some ideas from a widely used tool to visually display local alignments between pairs of sequences, namely *dot-plots*. In particular, we observed that aligning a sequence with itself, TRs form a regular pattern. Our algorithm mimics such visual search for these patterns to accomplish TR discovery. One of the main novelties of our approach is a compact representation of the dot-plot matrices that: (i) allows us to scale at genome-wide analysis, and (ii) can find application to other problems where dot-plots are used. Our algorithm belongs to the class of *ab-initio* methods, it allows both a tunable degree of divergence from the consensus sequence and a small insertion between two consecutive motifs. Both fasta and fastq are accepted as input allowing the analysis of NGS sequences. *Dot2dot* implements an optional customizable filter able to: remove biologically irrelevant results, and control the degree of overlap among TRs in the output list. Under the sensible assumption that the longest TR in the input sequence is much shorter (by orders of magnitude) than the entire sequence, our algorithm runs in linear time, thus enabling the analysis at whole-genome scale. Besides designing a new searching algorithm, we built five testing datasets covering diverse applicative areas. In particular: we collected from several public sources a set of 45 validated pathology-linked TRs; we compiled the list of coordinates of the CODIS loci including the seven loci that have been added since January 2017; we mapped on the hg38 reference genome a set of 620 manually annotated TRs reported in the Marshfield panel of variable loci; and we computed a catalog of 15 326 TRs located in upstream regulatory regions.

2 Materials and methods

Our algorithm leverages on a data structure at the base of dot-plots. Dot-plots are used to gain a visual insight of local alignments between two different sequences or even a sequence against itself. Matches between two elements are represented as (typically black) spots. A natural extension of this visual representation of alignments allows the use of color graduation to represent degree of similarity between pairs of elements (Sonnhammer and Durbin, 1995). The underlying data structure (called dot matrix) is a matrix where the element $M[i, j]$ stores the degree of similarity between the character in position i of the first sequence and the element in position j of the other string. When a sequence s is aligned with itself, $M[i, j]$ stores the degree of similarity between the element in position i and that in position j of s.

We observed that TRs form a distinctive pattern on self-sequence alignment dot-plots and, in turn, this pattern reflects on the underlying dot matrix. Consider e.g. a pure TR, since each instance of the motif perfectly aligns with the first instance, it will form a diagonal on the dot-plot. All these diagonals will lie stacked over the main diagonal. Counting the number of stacked diagonals we compute the number of copies of the consensus sequence, while from the length of the diagonals we derive the motif length. Fuzziness of TRs can easily be captured within this model. In fact, mismatches correspond to gaps in the diagonals, deletions cause interrupted diagonals and insertions cause the shift of the remaining part of the TR (see examples in the Supplementary Material). We noticed that the presence in the dot matrix of the above described pattern is a necessary and sufficient condition for the existence of a tandemly repeated sequence in the input, thus an algorithm that locates all and only these patterns ensures a computationally correct and complete tool for TRs discovery.

The naive quadratic cost of building, storing and searching the dot matrix is inadequate for whole-genome analysis. In order to lower the memory consumption and speed up the computation we propose an alternative representation of this data structure that can be built and stored in linear time/space. We also propose a fast searching heuristic algorithm to enumerate all the instances of the TR pattern in the dot matrix.

2.1 Data structure

In this section we provide details on how to infer the dot matrix M without explicitly building it. Let $S = s_1 s_2 \dots s_n$ be a sequence of length n over a finite alphabet Σ (where $\Sigma = \{A, C, G, T\}$ in our case). Given a character $x \in \Sigma$ we define $P(x)$ as the set of positions of S where $s_i = x$. The comparison of the character x and the sequence S induces a row $M[x]$ of M whose content does not depend on the position of x in the sequence but only on the content of S. As a consequence, comparing the sequence S with itself, the resulting matrix M has $|P(x)|$ rows identical to $M[x]$.

Following the above observation, building the whole M it suffices to compute only the vector $M[x]$ for each $x \in \Sigma$. In order to obtain a direct positional access to the induced dot matrix we build an auxiliary vector V of length n where in position i we store a reference to $M[s_i]$. Figure 1 shows an example data structure.

Since $|\Sigma|$ is a constant much smaller than n, building the data structure inducing M has time/space cost linearly proportional to n. In fact, both V and the $M[x]$s are vectors of size n that can be filled with a single scan of the sequence S.

Fig. 1. Sample data structure for the matrix associated to the sequence TTACGACGTACGATGACGACGT

2.2 TR searching procedure

Searching is a relatively simple procedure that scrolls a window of fixed length along the sequence and, for each position, checks for a stacked diagonal in the matrix M.

Let $D_M(i,j,l) = \{V[i][j], V[i+1][j+1], \ldots, V[i+l-1][j+l-1]\}$ be a l-long sequence of adjacent cells along M starting from position (i, j) such that two consecutive elements have coordinates that differ by 1 in terms of both rows and columns and let $W_M(i,j,l) = \sum_{k=0}^{l-1} V[i+k][j+k]$ be the summation of the scores of M over the positions $D_M(i,j,l)$. $D_M(i,j,l)$ is a diagonal if and only if $W_M(i,j,l) > l - \delta$ where $\delta \in [0,l]$ is a user-defined threshold. Given a certain position i on the sequence S, the event $W_M(i,i+l,l) = l$ corresponds to a pure TR of motif length l and copy number at least 2 starting from position i. Again $W_M(i,i+l,l) = W_M(i,i+2l,l) = l$ corresponds to a pure TR with copy number at least =3 and so on. The parameter δ is used to control the degree of fuzziness of TRs. In fact, the event $W_M(i,i+l,l) = l - \delta$ means that the second copy of the repeat contains δ mismatches. In its simplest form, the core of our searching procedure (depicted in Algorithm 1) scans the sequence S (see line 1) and checks for different values of l (see line 2) if the event $W_M(i,i+l,l) \geq l - \delta$ is verified. When this happens, the algorithm iteratively attempts to locate further diagonals in positions $i+2l, i+3l$ and so on (see lines 7–10). The procedure stops when an integer c such that $W_M(i,i+cl,l) < l - \delta$ is found. The sequence $S[i,i+cl]$ is reported as a TR with motif length l and copy number c.

We further refined our procedure to deal with insertions and deletions. We restrict to the most significant class of these variants. In particular, insertions can occur only between two copies of the motif sequence and their length is limited to be lower than the motif length l (see line 12). In addition, the length of insertions is fixed within the same TR. This means that, if a TR contains two or more insertions, they must have the same length to be correctly detected. Deletions are modeled as the insertion of a spurious sequence between two copies of the motif string. Although this model can appear limited, it has practical advantages and it is consistent with the replication slippage process described in Viguera *et al.* (2001). In fact, without a limit on the length of an insertion nearly every genomic sequence can be confused with a TR. The constraint on the equality of the insertion lengths within the same TR helps to predict the correct motif length and copy number when dealing with impure TRs. From the biological point of view, according to the model described in Viguera *et al.* (2001), the polymerase is arrested after replicating a unit of the repeat. Then, the realignment between the new strand and the template causes the insertion/deletion to happen between two copies of the TR motif sequence. In presence of an insertion between two copies of the consensus motif the condition $W_M(i,i+cl,l) >= l - \delta$ becomes false for a certain c. In this case (see lines 12–16), we seek for a gap checking for a possible diagonal in at most the next $l-1$ positions (i.e. $i+cl+1$, $i+cl+2$, etc.). In case of success we take note of the gap and its length and continue the standard searching procedure checking the next diagonal at distance l. When the condition on $W()$ becomes false again we do not test all the possible sizes of the gap but only the previously annotated length.

In order to enable *Dot2dot* to identify the longest possible TR, the searching procedure still has to check whether the sequence immediately downstream a TR is a prefix of the consensus sequence, even though it does not satisfy the condition $W_M() < l - \delta$. In this case, however, we seek only for perfect matching so as not to reduce TR's purity. We perform this test iteratively verifying (see lines 22–25) that the i-th character in the downstream sequence matches the i-th character of the motif (i.e. the first instance of the TR). If a suitable prefix is found, it is included in the output TR.

In terms of asymptotic analysis, the overall computational cost of *Dot2dot* is proportional to the number of times the condition $W() > l - \delta$ is tested. A single computation of $W()$ takes $O(l)$ time since it costs l accesses to the matrix M. Given a certain position $p \in [1, |S| - 2l]$ of S, there are two cases: either there exists in S a TR with k_p copies starting in position p or not. In the first case $W()$ is computed $k_p + l$ times while in the latter case $W()$ is computed only once. In general, however, l is constant, but k_p cannot be bounded and, thus, it can hold $k = \max_{p \in [1,|S|-2l]}(k_p) = |S|/l$ in the worst

Algorithm 1 Tandem repeat searching procedure

Require: sequence S, dot matrix M, parameter δ
Ensure: list R of tandem repeats
1: **for all** $i \in [1, |S|]$ **do**
2: **for all** $l \in$ *range_of_motif_lengths* **do**
3: $cn \leftarrow 1$;
4: $gap\_offset \leftarrow gap \leftarrow 0$;
5: **repeat**
6: *can_extend* $\leftarrow$ **false**;
7: **while** $W_M(i; i + gap\_offset + (cn * l); l) < \delta - l$ **do**
8: $cn \leftarrow cn + 1$;
9: *can_extend* $\leftarrow$ **true**;
10: **end while**
11: **if** $gap = 0$ **then** try to guess gap length
12: **for** $k \leftarrow 1; gap == 0$ **and** $k < l; k \leftarrow k + 1$ **do**
13: **if** $W_M(i; i + gap\_offset + k + (cn * l); l) < \delta - l$ **then**
14: $gap \leftarrow k$;
15: **end if**
16: **end for**
17: **end if**
18: $gap\_offset \leftarrow gap\_offset + gap$;
19: **until** *can_extend* is **true**;
20: **if** $cn \geq 2$ **then** Found a valid TR
21: $gap\_offset \leftarrow gap\_offset - gap$;
22: $last \leftarrow 0$;
23: **while** $last < l$ **and** $S[i + last] = S[i + gap\_offset + (cn * l) + last]$ **do**
24: $last \leftarrow last + 1$;
25: **end while**
26: R.append $(S[i, i + gap\_offset + (cn * l) + last]$, copy_number $= cn$, motif_length $= l)$;
27: **end if**
28: **end for**
29: **end for**

case. Moreover, a similar condition can hold for every position p. As a result, the overall running time of the searching procedure would be $|S|lk = O(|S|k) = O(|S|^2)$ in the worst case.

In terms of average-case analysis, we have to estimate the value $\hat{k} \leq k$ that balances the high cost paid every time a new TR is found and the low cost paid otherwise. Dealing with real genomic sequences we observed that the probability for a given position to be the starting point of a TR is fairly low. Moreover, the longest TR is order of magnitude shorter than the input sequence. These assumptions would lead to an expected linear running time of our algorithm. To confirm our hypotheses we estimated the value of $\hat{k}$ over the entire hg38 reference genome. According to our experiments we measured $k = 300$ and $lk = 2495$. Moreover, the probability of a random position p to be the starting coordinate of a TR $=0.0051$. Consequently, the expected value of $\hat{k}$ is 1.53.

2.3 Filtering

Exhaustive approaches to TR discovery suffer from the fact that many reported results may be artifacts rather than proper TRs. For example, finding a TR of copy number c, this class of algorithms returns also all the sub-instances of copy number $c-1$, $c-2$, etc. Since this behavior has also a strongly negative impact on running time, it would be better avoiding computing these artifacts instead of filtering them a posteriori. In order to solve this problem, we maintain a bit vector to mask those positions that will certainly produce such an undesired result. Once our searching procedure identifies a new TR, it sets the bit corresponding to the first position of each copy of the motif sequence. Marked positions are ignored during the subsequent searching. We notice that this filter applies not only to pure TRs, but its benefits partially extend to fuzzy TRs. Suppose e.g. that we want to find TRs within the toy sequence ACG AGG ACG ACG ACG AGT AGT and let suppose $\delta = 1$. In this case, *Dot2dot* would return the TR ACG AGG ACG ACG ACG which is not pure because of the substitution C/G in the second copy of the motif. However, our filtering would avoid computing two uninteresting TRs consisting in: the last three instances of the motif (ACG ACG ACG), and the last two instances (ACG ACG). A tandem beginning with the motif AGG, instead, needs to be searched since it could be the starting point of a second TR not entirely included in the first result (in the case of the example AGG ACG ACG ACG AGT AGT).

A second class of artifacts is that of pairs (or groups) of overlapping results consisting in the same TR (with the same motif length and copy number) shifted one position forward along the sequence (namely starting and ending positions differ by only 1 bp). This case arises only when the initial characters of the TR match the corresponding characters immediately after the TR. Dealing with this class of artifacts, the searching procedure attempts to expand a retrieved TR allowing it to have a final (fractional) pure motif (see lines 22–25 of Algorithm 1). The resulting extended TR is conceptually equivalent to merging together a sequence of shifted TRs into a longer result. The above equivalence suggests a sufficient condition to determine whether searching from a given position would lead to find an artifact. In fact, since a TR with a $0 < \hat{l} < l$ long final motif is equivalent to a sequence of $\hat{l}$ TRs in which motifs are shifted by 1 bp, all the positions at distance at most $\hat{l}$ from a marked element in the bit vector must be the starting coordinates of a sub-TR of the expanded one. As a result, these positions can be ignored.

A last class of artifacts derives from the fact that the same TR can admit several distinct combinations of period length and copy number. Since *Dot2dot* increasingly tests several possible motif lengths (see line 2 of Algorithm 1), the absence of a specific mechanism to filter (or to avoid the computation of) all the possible combinations of the same result would lead to a potentially huge redundancy in the output. Dealing with impure TRs the only available option is that of computing all the combinations and then evaluating which one better fits a pre-determined criterion (in our case we prioritize the purest one). Pure TRs, instead, do not require computing all the possible combination. In fact, testing for multiples of the motif length can only produce either TRs of the same size or shorter. Consequently, once we find a pure TR we do not need to test multiples of its motif length.

3 Results

We experimentally tested our software to assess whether it is able to find non-naive biologically relevant TRs. Moreover, we thoroughly scanned the literature to find the widest possible pool of alternative algorithms to compare with. As a testing dataset we used five collections of TRs: 45 disease-related TRs, the 20 extended CODIS repeats, a set of Y-STR loci, 620 markers from the Marshfield panel and a wide list of TRs located in the regulatory regions. Although it could be considered inconsequential for our purposes, we choose to map all TRs and genes onto the most recent main release of the human genome (namely hg38). We manually mapped genes querying the UCSC genome browser (https://genome-euro.ucsc.edu) while we used the batch *LiftOver* interface (https://genome.ucsc.edu/cgi-bin/hgLiftOver) to convert the coordinates of TRs. Results show that our method is able to find biologically relevant repeats not reported from the other methods.

3.1 Tandem repeats discovery tools selection

Despite the vast literature on TRs discovery tools [see Lim *et al.* (2013) for a recent survey], only few algorithms are usable in practice. In fact, most tools seem to be no longer available or no longer supported (Abajian, 1994; Krishnan and Tang, 2004; Kurtz *et al.*, 2001; Parisi *et al.*, 2003; Pokrzywa and Polanski, 2010; Sokol *et al.*, 2007; Taneda, 2004; Wirawan *et al.*, 2010; Zhou *et al.*, 2009); some other is still available but no longer maintained [this is the case e.g. Wexler *et al.* (2005)] that is distributed only in binary form and requires an obsolete version of the operating system). Some algorithms are subjected to limitations that make comparing with them unfair. In particular: STAR (Delgrange and Rivals, 2004) uses a dictionary-based approach that makes its computational cost unaffordable even using a small dictionary; MsDetector (Girgis and Sheetlin, 2013) and IMEx (Mudunuri and Nagarajaram, 2007) are designed only for microsatellites with motif length ≤ 6; the approach in Thiel *et al.* (2003) leverages on a species-specific database; E-TRA (Karaca *et al.*, 2005) can only find perfect TRs; and Pop (2015) provides only a visual representation of the distribution of TRs over the input sequence.

At the end of our investigation we identified only seven algorithms that can be realistically employed in daily TR discovery tasks: tandem repeats finder (TRF) (Benson, 1999), mreps (Kolpakov *et al.*, 2003), tandemSWAN (Boeva *et al.*, 2006), TRStalker (Pellegrini *et al.*, 2010), SciRoKo (Kofler *et al.*, 2007), TROLL (Castelo *et al.*, 2002) and RepeatMasker (Smit *et al.*, 2017) (http://www.repeatmasker.org). Since all these methods are provided with a default set of parameters that perform well in most cases, after verifying that they match the characteristics of our datasets, we always used them (see details in the Supplementary Material).

3.2 Disease-associated tandem repeats

Due to the proven relationship between repeat expansion and a consistent number of neurological and neuromuscular disorders

(Mirkin, 2007), we tested our algorithm with the aim of evaluating its ability to locate significant TRs associated with diseases. To obtain a convincing list of pathology-related TRs we built our dataset collecting information from several sources. In Castel *et al.* (2010) a list of 44 well-known polymorphic TRs (36 of which confirmed to be associated with a pathology) is given. We removed from this list two entries: SCA31 because the associated repeat is not present in the reference genome since the Spinocerebellar Ataxia type 31 is caused by the insertion of the entire repeat (Sato *et al.*, 2009), and Facioscapulohumeral muscular dystrophy because the 3.3 k D4Z4 macrosatellite repeat is too long for all the tested algorithms. Due to the lack of the chromosomal coordinates in the above list, we extracted this information from other sources. In a master thesis (Mador-House, 2014) of the same research group of the article in Castel *et al.* (2010), the authors extend the list including four new TRs (two of which linked with a pathology: C9ORF72 and SCA36) and removing two [one linked with the fragile X tremor/ataxia syndrome and one (FRA16A) not directly linked with a specific pathology] providing the coordinates of the TRs in the hg19 reference genome. From the list in Mador-House (2014) we removed 12 elements corresponding to TRs with not confirmed association to a pathology (at the time of this publication) because (as the author mentioned in the thesis) they have been located using the Tandem Repeat Finder software. Since some of the TRs in Castel *et al.* (2010) have now been confirmed as associated with a pathology, and thus their sequence and genomic position is known, we could manually extend our list exploiting *blastn* (http://blast.ncbi.nlm.nih.gov) to locate them.

In particular: according to Todd *et al.* (2013) the sequence of the TR causing the fragile X tremor is an almost pure CGG sequence; in Wieben *et al.* (2012) the authors provide the sequence of a repeat expansion in the transcription factor 4 (TCF4) causing the Fuchs corneal dystrophy; in DeJesus-Hernandez *et al.* (2011) the authors report the motif of a pure TR whose expansion in the non-coding region of C9ORF72 is associated with FTD and ALS; and in Grube *et al.* (2011) the trinucleotide expansion in KCNN3 reported in Chandy *et al.* (1998) appears to be associated with schizophrenia. We further extended our list including other notable TRs reported in the literature. In Pellegrini *et al.* (2012) the authors list 29 TRs two of which (one in PHOX2B and one in SOX3) were not included in our list. In Winnepenninckx *et al.* (2007) an expansion of a pure CGG-repeat in the 5' UTR of the DIP2B gene is associated with the FRA12A disease. In de Pontual *et al.* (2003) a mutation of the proneural HASH-1 gene is associated with CCHS. Finally, a CAG repeat in POLG1 has been associated with male infertility in Aknin-Seifer *et al.* (2005) and recently with breast cancer risk in Azrak *et al.* (2012). Our final dataset consists in 45 TRs with: a disease-associated polymorphism, period ranging from 3 to 24 bp, and size ranging from 15 to 405 bp (see Supplementary Material for the complete list and coordinates).

3.3 DNA profiling tandem repeats

As a consequence of their contribution to DNA profiling in forensic sciences, a large database of short TRs [STRbase Ruitberg *et al.* (2001)] has been made publicly available on the web (http://www.cstl.nist.gov/strbase/). Although STRbase does not report the exact genomic location of the censused STRs, it provides useful information that we exploited to pinpoint the coordinates of the two most commonly used collections of TR listed in it: CODIS and Y-STR.

3.3.1 FBI CODIS

The Combined DNA Index System (CODIS) database consists of 13 tetra-nucleotide TRs spread in 12 chromosomes. As of January 1, 2017 this dataset has been extended with 7 new tetra-nucleotide TRs: D1S1656, D2S441, D2S1338, D10S1248, D12S391, D19S433 and D22S1045. We collected the coordinates in hg19 of the 13 CODIS core STR loci from the lobSTR (Gymrek *et al.*, 2012) website, while we manually retrieved the coordinates of the new TRs.

3.3.2 Y-STR

Y-STR is a collection of short TRs in the *Y* chromosome with period lower than 6 and size ranging from 24 to 166 bp. These repeats are often used for paternity or genealogical tests as well as for forensic purposes. An almost complete list of these loci has been published in Gymrek *et al.* (2013) and the genomic coordinates in hg19 are available in the lobSTR website. The two markers DYS448 and DYS449 are not TRs in a strict sense and consist of two pure STRs interrupted by a small spurious sequence. According to Gymrek *et al.* (2013), for these markers we included both: the entire STR and the two parts. We completed our list including two extra TRs not mentioned in Gymrek *et al.* (2013) but present in the lobSTR website: DYS640 and DYS464. The forensic value of DYS464 is studied in Butler and Schoske (2004). The final collection of Y-STR consists of 86 loci.

3.4 Marshfield linkage panel

The Marshfield linkage panel (Rosenberg *et al.*, 2005) consists of more than 600 loci distributed across the autosomes each of which containing a short and highly polymorphic TR. The main purpose of this panel is that of exploiting the great degree of polymorphism of TRs to conduct genome-wide population analyses. The original work in Rosenberg *et al.* (2005) neither describes in details the structure of the TRs nor reports their coordinates. Thus, some authors used TRF to pinpoint the repetitive structures. This choice is sensible for certain problems [e.g. in Willems *et al.* (2017) where the Marshfield panel is used as a benchmark for genotypization], but it is inappropriate for our purposes. In the Supplementary Material of Pemberton *et al.* (2009), the authors provide the PCR primers, the RefSeq sequences, as well as a manually curated description of the TR structures, for 627 of the Marshfield loci (see the file Pemberton_AdditionalFile1_11242009.txt in the Rosenberg's website https://web.stanford.edu/group/rosenberglab/data/pembertonEtAl2009/). Using the RefSeq sequences as input for *blastn* (http://blast.ncbi.nlm.nih.gov) we located 598 loci. We further obtained the hg38 coordinates of another 22 loci by means of the UCSC's *in silico* PCR tool (https://genome.ucsc.edu/cgi-bin/hgPcr). Finally, we pinpointed the exact coordinates of each TR by a local alignment of the flanking regions of the TR in refseq and the locus in hg38. The resulting dataset consists in 620 microsatellites with size ranging from 8 to 270 bp.

3.5 Tandem repeats in the regulatory regions

Despite being ubiquitously distributed in eukaryotic genomes, TRs have been reported to be more abundantly present in regulatory regions and in particular in promoters (Sawaya *et al.*, 2013; Vinces *et al.*, 2009). Their great variability as well as intrinsic instability suggests that mutations of this class of genotypic variation in the promoter regions can influence the observed phenotype (Gemayel *et al.*, 2010). Notwithstanding their importance, a manually curated reference database of variable TRs in promoter regions is not available yet. Partial lists of pure TRs are reported in Heidari *et al.* (2012) and Ohadi *et al.* (2012) while in Bolton *et al.* (2013) the authors describe a resource where a partial list is obtained by means of the only TRF software. Besides the database resource, the authors of Bolton *et al.* (2013) provide a description of a sensible

methodology that we used to recompute the list of TRs in the promoter regions using all the algorithms available to us.

Following the procedure in Bolton *et al.* (2013) we downloaded the coordinates of the human genes from the UCSC table browser (Karolchik *et al.*, 2004), then we removed non-coding and putative genes as well as genes not in haplotypic regions. After removing duplications we obtained a list of 36 096 elements including coding genes and isoforms. In order to ensure that the promoter regions have been entirely covered, as an input sequences we used a 3 kb interval from −2 kb to +1 kb from the transcription start sites. We run all the algorithms on these sequences limiting the TR period up to 9 bps, discarding results with purity lower than 90% and removing TRs shorter than 25 bp. Given the reasonably limited number of results, we could refine the list of TRs by: re-estimating the correct period length, and trimming spurious endpoints. Estimating the period length, we used a brute force procedure that tested all the possible combinations and decided for the shortest value maximizing purity. Then we compared the first and the last instance of each TR motif with the corresponding consensus sequence and trimmed those not matching. Finally, we get rid of overlapping TRs by means of an iterative procedure that, at each step, removed the less pure overlapping TR or, in case of tie, the overlapping TR with higher motif length or the shortest one. At the end of this procedure we obtained a final list of 15 326 TRs.

3.6 Evaluation metrics

Assessing quality of TR discovery algorithms via comparing results with a gold standard requires a few caveats. A first issue is the definition of matching. In the most restrictive setting, one could be interested in the exact match of the starting and ending coordinates, while in general a limited degree of divergence is acceptable. A perfect match is very hard to achieve and it could not be significant because of the subjectivity of the identification procedure. In fact, the coordinates of (more often impure) TRs can slightly differ according to the interpretation given by the curator during the manual annotation phase. In addition, when a TR is adjacent to a region similar enough to the repeat itself, many alternatives are equally possible. In this case, establishing which is the 'correct' repeat, results in an arbitrary choice.

Another relevant issue is that of redundancy (namely, the presence in the output of overlapping TRs). Despite often due to software artifacts, a moderately redundant output may not be problematic when the subsequent analysis is automated, while it could defeat the purpose of an algorithm when only a restricted number of results can be analyzed. An evaluation based only on the score of the best hit can penalize those algorithms that employ filtering to reduce the number of biologically irrelevant results.

Finally, the possibility of defining the concepts of true/false positives/negatives, that are at the base of measures like sensitivity and specificity, should be critically examined. As observed in Saha *et al.* (2008), evaluating an *ab-initio* method through a gold standard dataset (either the output of another algorithm or a collection of TRs) the true positives are easily defined as the algorithm results that are also listed in the gold standard, while the false negatives are sequences of the testing dataset not reported in the list of results. The problem arises with the other two classes. In particular, it is questionable whether a result reported by the tool but not present in the reference dataset is a false positive or it is a new legal TR that was not previously known. The difficulty of defining the concepts of false positives and true negatives has the effect of hindering the estimation of specificity. Sensitivity, instead, can be computed through the standard formula. However, since sensitivity and specificity are dual evaluation functions that need to be considered as a whole, we decided to use alternative measures. In particular, we used precision and recall.

In order to address the issue of matching algorithm results with the gold standard, we used the Jaccard coefficient. This score has originally been employed to measure the degree of similarity between sets. Subsequently, it has been extended to measure the overlap between intervals. In short, the Jaccard coefficient is defined as the ratio of the length of the intersection of two intervals divided by the size of their union. Its value is bounded in the range $[0, 1]$ and to a higher value corresponds a higher degree of matching. Entirely covering a TR is necessary but not sufficient to get the highest score, in fact, the Jaccard coefficient has value 1 only when the comparing intervals have exactly the same coordinates.

Dealing with redundancy, we used three measures: the average number of results covering an element of the gold standard, the average precision and the average recall. Let $T = \{t_1, \ldots, t_n\}$ be a dataset of TRs, R be the set of results of a given algorithm, and $R(t_i)$ be the subset of R overlapping with t_i by at least 1 bp. We further denote $jac()$ as the Jaccard coefficient between two genomic intervals. The average precision and average recall are defined as follows:

$$\sigma_P(T,R) = \frac{1}{n}\left(\sum_{i=1}^{n}\frac{\sum_{x\in R(t_i)} jac(x,t_i)}{|R(t_i)|}\right)$$

$$\sigma_R(T,R) = \frac{1}{n}\left(\sum_{i=1}^{n}\max_{x\in R(t_i)} jac(x,t_i)\right).$$

The above measures have several advantages. Firstly, they are independent from the arbitrary choice of a threshold value. Secondly, results over different datasets can be merged into a single score or can be compared directly. Lastly, it is possible to compare algorithms that apply filtering with methods that do not use it. In fact, even in a case where the filtering phase removes the most overlapping repeat, causing the Jaccard score of the second best hit to drop under threshold, the removal of a promising repeat could be balanced by the removal of a certain number of irrelevant repeats with low Jaccard score.

4 Discussion

In this section we discuss the outcome of our comparative evaluation. We run *Dot2dot* using two sets of parameters: one with the default values but without any filtering, and one with the same values but enabling the most stringent filtering (see Supplementary Material for details). TRStalker was run in multiprocessor mode using 64 threads. We also set a time limit of four days to accomplish a single run over a sequence. Because of the severe restrictions on the input length due to the computational cost of TRStalker, we run this software also on trimmed subsequences of length at most 10 kbp centered on a TR (notice that this had no effect on the TRs in the regulatory regions since the input is always shorter). Although a direct comparison of TRStalker on the diseases and CODIS datasets with or without shortened input reveals a remarkable advantage in favor of the former (henceforth indicated asTRStalker*), using trimmed input was the only way to estimate TRStalker performances on all the datasets.

Table 1 reports the number of TR identified setting Jaccard score to 0.5 and 0.7 (see results for other thresholds in the Supplementary Material), the average number of results per locus (RPL), the average precision and the average recall of each algorithm for the five test collections. Accepting *Jaccard* =0.5 *Dot2dot* and TRStalker* rank alternatively first and second in terms of absolute number of

Table 1. Average precision σ_P, average recall σ_R, average number of reported results covering a target locus (RPL) and total number of TRs intersected (with Jaccard=0.5 and Jaccard=0.7) of the compared algorithms and datasets

Dataset	Measure	Dot2dot-filter	Dot2dot	TRF	MREPS	TRStalker	TRStalker*	SWAN	Troll	SciRoKo	Repeatmasker
Diseases	σ_P	**0.830**	0.678	0.722	0.630	0.464	0.491	0.372	0.508	0.724	0.452
	σ_R	0.835	0.899	0.763	0.686	0.901	**0.960**	0.575	0.571	0.752	0.475
	RPL	**1.0**	6.0	1.4	1.5	155.9	157.5	2.0	4.1	1.1	1.2
	#TR j=0.5	44	**45**	37	37	43	**45**	27	29	38	24
	#TR j=0.7	36	42	33	23	43	**45**	24	18	33	20
CODIS	σ_P	**0.860**	0.721	0.836	0.659	0.485	0.565	0	0.682	0.819	0.812
	σ_R	0.860	0.899	0.850	0.818	0.839	**0.988**	0	0.797	0.867	0.822
	RPL	**1.0**	6.3	1.1	2.1	188.2	172.2	0.0	3.2	1.2	**1.0**
	#TR j=0.5	19	19	**20**	**20**	18	**20**	0	19	**20**	19
	#TR j=0.7	18	18	15	15	18	**20**	0	16	16	15
Y-STR	σ_P	**0.877**	0.706	0.770	0.617	—	0.535	0.021	0.682	0.827	0.721
	σ_R	0.879	0.916	0.786	0.712	—	**0.979**	0.029	0.777	0.841	0.729
	RPL	**1.0**	6.1	1.1	2.1	—	205.5	2.5	2.8	1.1	**1.0**
	#TR j=0.5	84	**86**	70	74	—	**86**	3	79	80	67
	#TR j=0.7	74	81	65	59	—	**86**	1	75	69	62
Marshfield	σ_P	**0.856**	0.637	0.784	0.662	—	0.559	0	0.683	0.810	0.767
	σ_R	0.860	0.905	0.794	0.755	—	**0.975**	0	0.792	0.830	0.775
	RPL	**1.0**	7.1	1.1	2.0	—	154.3	0.0	3.2	1.1	**1.0**
	#TR j=0.5	589	609	559	577	—	**619**	0	583	587	554
	#TR j=0.7	503	557	447	405	—	**619**	0	490	471	437
Promoters	σ_P	**0.825**	0.667	0.663	0.575	0.422	0.422	0.663	0.517	0.808	0.743
	σ_R	0.803	**0.934**	0.548	0.687	0.929	0.929	0.447	0.627	0.709	0.432
	RPL	1.0	4.8	1.4	1.9	116.0	116.0	2.2	2.1	1.0	**0.6**
	#TR j=0.5	13 783	15 192	8864	12 490	**15 264**	15 264	5156	10 939	12 125	7128
	#TR j=0.7	12 329	14 792	7905	7880	**15 098**	15 098	3233	5723	10 492	6286

Note: TRStalker run on the trimmed sequences is reported as TRStalker *. (The best value is highlighted in bold).

located TRs with a negligible gap. TRStalker on the diseases loses only two TRs, while the other algorithm in general misses a remarkable number of elements. Increasing Jaccard to 0.7 causes *Dot2dot* to became less accurate than TRStalker and TRStalker* but still more accurate than the others. This higher accuracy, however, is the effect of a very large redundancy due to the exhaustive enumeration of all the possible alternative TRs covering the same locus. In fact, TRStalker and TRStalker cover a locus with an average of over 110 different results.

Although the ability of identifying the repeats of the gold standard with a reasonably high Jaccard score is a desirable property, achieving a high recall at the cost of a large output redundancy can greatly reduce the usefulness of an algorithm in practice. In particular, a good filtering strategy should increase the average precision at the cost of a negligible reduction of accuracy.

Table 1 shows that an aggressive filtering typically has a positive impact on the average precision as well as on controlling the redundancy of the output. In fact, the algorithms implementing the most aggressive filtering strategy (*Dot2dot-Filter*, TRF and SciRoKo) achieve the highest values of average precision. As a notable exception, the (embedded) purity-based filtering of TandemSWAN caused the algorithm to filter out most of the TRs on the profiling datasets because they were considered uninteresting. A high level of redundancy is not necessarily a guarantee of a high average recall. This is the case, e.g. of Troll that has the third highest number of RPL but achieves one of the lowest average recall. *Dot2dot*, TRStalker and TRStalker*, which have the most verbose output, achieve the highest average recall. However, in spite of having comparable average recall (almost the same narrowing to TRStalker), *Dot2dot* returns a number of RPL of the same order of magnitude of the other methods, while TRStalker produces two orders of magnitude more RPL.

Comparing *Dot2dot* run with and without filtering, we observed how our filtering strategy has a remarkably positive impact in terms of a drastic reduction of overlapping results, passing from about 5 RPL to ∼1. Filtering causes an increase of average precision of about 0.15 allowing *Dot2dot* to attain the highest score over all the tested datasets at the cost of a limited reduction of average recall (−0.06). However, this lower recall does not change the relative ranking of our algorithm in comparison with the other methods.

4.1 Running time

Although running time is a secondary feature for TRs discovery algorithms, it can still be important to evaluate the feasibility of large-scale analyses of whole-genomes and NGS data. Assessing the selected algorithms on whole-genomes, we run them on seven assembled references with sizes ranging between 1.3 and 27 Mbp, available on the NCBI website. NGS data, instead, have been tested on the high-coverage long reads of the NA12878 individual sequenced with Oxford Nanopore Technology (Jain *et al.*, 2018). However, since *Dot2dot* and TRF are the sole two methods that can directly handle large multifasta files (Mreps, Tandem SWAN can handle only one sequence at time, Troll and SciRoKo, instead, require one file for each sequence to be specified in the command line). We narrowed comparison on NGS data only to them.

Table 2. Running time of the compared algorithms over seven common reference genomes

Organism name	Size Mb	Dot-to-dot filter	TRF	Mreps	Tandem SWAN	SciRoKo	Troll
Pinus lambertiana	27602.70	12 h 15 min 18 s	16 h 15 min 14 s	N/A	N/A	2 h 13 min 51 s	N/A
Triticum aestivum	13427.40	5 h 57 min 14 s	11 h 13 min 43 s	N/A	N/A	41 min 59 s	N/A
Locusta migratoria	5759.80	2 h 37 min 14 s	2 h 25 min 12 s	11 h 20 min 14 s	N/A	18 min 39 s	N/A
Homo sapiens	3241.95	1h 24min 15s	2 h 40 min 51 s	2 h 36 min 00 s	12 h 31 min 48 s	10 min 43 s	35 min 14 s
Rattus norvegicus	2870.18	1 h 18 min 52 s	2 h 04 min 44 s	N/A	N/A	9 min 40 s	N/A
Mus musculus	2807.72	1 h 16 min 29 s	1 h 55 min 37 s	N/A	N/A	9 min 41 s	N/A
Danio rerio	1371.72	36 min 29 s	2 h 33 min 36 s	2 h 07 min 54 s	13 h 59 min 40 s	5 min 7 s	17 min 16 s
NA12878	82705.67	1 d 12 h 15 min 18 s	1 d 16 h 15 min 14 s	—	—	—	—

Note: The genome dimension is measured as the size in Mb of the corresponding fasta file.

We used a MacOS-based workstation endowed with a processor Intel Xeon 3.1Ghz and run the software in single thread mode. For tandemSWAN and Troll, which run only under Linux, we had to use a different hardware. We computed the performance difference between the two machines by comparing the speed of *Dot2dot* over chromosome 1 of hg38. We found that the Linux server is 1.4556 times faster. As a result, we used this constant to adjust tandemSWAN and Troll's running time.

We report in Table 2 the running time of the tested algorithms. We excluded TRStalker because it cannot run on large sequences and Repeat Masker because its running time is dominated by the identification of other classes of repetitions different from TRs. Since we endorse filtering, we run *Dot2dot* only with it. We notice that this choice does not give *Dot2dot* any advantage in the comparison, in fact, without filtering our software would run slightly faster.

As Table 2 shows, three algorithms are unable to run on the majority of the genomes. This is mostly due to intrinsic thresholds hardwired in the software. In particular, Mreps has a constraint on the length of the longest consecutive stretch of Ns in a sequence, while Troll, which would be the second fastest software, has a limit on the overall length of the input that cannot exceed 2^{31} bp (about 2.1 Gbp). Because of these limitations Mreps could not run on relatively small genomes like that of Mus musculus, while to run Troll on the human genome we had to search each chromosome independently and merge results together.

Being based only on the computation of the Hamming distance with a seed, SciRoKo consistently achieve the highest speed regardless the sequence length. However, the absence of any corrections to this simple mechanism causes SciRoKo to have one of the less rich output in terms of number of results per kilobase (as few as 0.61 TRs per kbp) with a large predominance of pure (or almost pure) TRs.

Despite the reachest output in terms of results per kBps (see details in the Supplementary Material), *Dot2dot* runs consistently faster than TRF. The gap between the two algorithms is quite negligible for small and medium-sized genomes, while it tends to became rather large for the two biggest assembled genomes. Comparisons on the NGS data, instead, show a substantial gap that can have a crucial weight in large-scale analyses. Moreover, the absence of a multithread release of the TRF software, and the consequent difficulty of using parallelism of modern CPUs, has the effect of sharpen this gap even more. In fact, running 24 parallel instances of TRF (one per chromosome), it still requires more than 22 h to finish while *Dot2dot* with 24 threads completes in 5 h.

Data on Table 2 confirm also the average-case cost analysis of our algorithm. In fact, the *Dot2dot* running time grows proportionally with the genome's length with a small stable constant factor.

5 Conclusion

In this paper we presented *Dot2dot*: a new algorithm for TR identification in a target genome. Our model of repeat has shown to be general enough to capture well various classes of TRs with different characteristics: pathology-linked, forensic, for population analysis, genealogic-oriented and repeats in the regulatory regions. Experiments have shown that *Dot2dot* is fast and effective since it was able to identify almost all the TRs of our test collections with an accuracy of at least 0.7 in terms of Jaccard score. Even applying a severely stringent filtering where overlap among the returned repeats is not allowed, our algorithm is still more accurate than the alternative tested tools. Tests over the entire human genome have confirmed the hypothesis that the longest TR (with a length of 2495 bp) is several order of magnitude shorter than any of the input sequences enabling *Dot2dot* to run in linear time with the input length.

Dot2dot is freely available and can be used without restrictions. To make it useful in the daily laboratory practice we enabled it to read standard formats for both assembled genomes (fasta) and NGS data (fastq) as well as return its output also in a standard bed format.

Another contribution of this paper is that of proposing a rigorous assessment methodology of TRs discovery algorithms as well as providing five reference collections of TRs (four of which manually curated). We hope that the proposed methodology and datasets can help to facilitate future research in this field.

Funding

This work was supported by the project RepeatALS FGBR 17/2013 funded by Arisla (Italian Society for Research on Amyotrophic Lateral Sclerosis) and the project PRIN 201534HNXC. The role of tandem repeats in neurodegenerative diseases: a genomic and proteomic approach funded by the Italian Ministry of Education and University (MIUR).

Conflict of Interest: none declared.

References

Abajian,C. (1994) Sputnik: DNA microsatellite repeat search utility.

Aknin-Seifer,I. *et al.* (2005) Is the cag repeat of mitochondrial dna polymerase gamma (polg) associated with male infertility? A multi-centre french study. *Hum. Reprod.*, **20**, 736–740.

Azrak,S. *et al.* (2012) Cag repeat variants in the polg1 gene encoding mtdna polymerase-gamma and risk of breast cancer in African-American women. *PLoS One*, **7**, e29548.

Bacolla,A. *et al.* (2008) Abundance and length of simple repeats in vertebrate genomes are determined by their structural properties. *Genome Res.*, **18**, 1545–1553.

Benson,G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res*., **27**, 573.

Boeva,V. *et al*. (2006) Short fuzzy tandem repeats in genomic sequences, identification, and possible role in regulation of gene expression. *Bioinformatics*, **22**, 676–684.

Bolton,K.A. *et al*. (2013) Starrrt: a table of short tandem repeats in regulatory regions of the human genome. *BMC Genomics*, **14**, 795.

Butler,J.M. and Schoske,R. (2004) Forensic value of the multicopy y-str marker dys464. *Int. Congr. Ser*., **1261**, 278–280.

Castel,A.L. *et al*. (2010) Repeat instability as the basis for human diseases and as a potential target for therapy. *Nat. Rev. Mol. Cell Biol*., **11**, 165–170.

Castelo,A.T. *et al*. (2002) Troll-tandem repeat occurrence locator. *Bioinformatics*, **18**, 634–636.

Chandy,K. *et al*. (1998) Isolation of a novel potassium channel gene hskca3 containing a polymorphic cag repeat: a candidate for schizophrenia and bipolar disorder? *Mol. Psychiatry*, **3**, 32–39.

de Pontual,L. *et al*. (2003) Noradrenergic neuronal development is impaired by mutation of the proneural hash-1 gene in congenital central hypoventilation syndrome (ondine's curse). *Hum. Mol. Genet*., **12**, 3173–3180.

DeJesus-Hernandez,M. *et al*. (2011) Expanded GGGGCC hexanucleotide repeat in noncoding region of C9ORF72 causes chromosome 9p-linked FTD and ALS. *Neuron*, **72**, 245–256.

Delgrange,O. and Rivals,E. (2004) Star: an algorithm to search for tandem approximate repeats. *Bioinformatics*, **20**, 2812–2820.

Gemayel,R. *et al*. (2010) Variable tandem repeats accelerate evolution of coding and regulatory sequences. *Annu. Rev. Genet*., **44**, 445–477.

Girgis,H.Z. and Sheetlin,S.L. (2013) Msdetector: toward a standard computational tool for DNA microsatellites detection. *Nucleic Acids Res*., **41**, e22.

Grube,S. *et al*. (2011) A cag repeat polymorphism of kcnn3 predicts sk3 channel function and cognitive performance in schizophrenia. *EMBO Mol. Med*., **3**, 309–319.

Gymrek,M. *et al*. (2012) lobstr: a short tandem repeat profiler for personal genomes. *Genome Res*., **22**, 1154–1162.

Gymrek,M. *et al*. (2013) Identifying personal genomes by surname inference. *Science*, **339**, 321–324.

Heidari,A. *et al*. (2012) Core promoter strs: novel mechanism for inter-individual variation in gene expression in humans. *Gene*, **492**, 195–198.

Jain,M. *et al*. (2018) Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat. Biotechnol*., **36**, 338.

Karaca,M. *et al*. (2005) Exact tandem repeats analyzer (e-tra): a new program for DNA sequence mining. *J. Genet*., **84**, 49–54.

Karolchik,D. *et al*. (2004) The ucsc table browser data retrieval tool. *Nucleic Acids Res*., **32**, D493–D496.

Kofler,R. *et al*. (2007) Sciroko: a new tool for whole genome microsatellite search and investigation. *Bioinformatics*, **23**, 1683–1685.

Kolpakov,R. *et al*. (2003) mreps: efficient and flexible detection of tandem repeats in DNA. *Nucleic Acids Res*., **31**, 3672–3678.

Krishnan,A. and Tang,F. (2004) Exhaustive whole-genome tandem repeats search. *Bioinformatics*, **20**, 2702–2710.

Kurtz,S. *et al*. (2001) Reputer: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Res*., **29**, 4633–4642.

Lim,K.G. *et al*. (2013) Review of tandem repeat search tools: a systematic approach to evaluating algorithmic performance. *Brief. Bioinform*., **14**, 67–81.

Mador-House,R. (2014) Investigation of the Epigenetic Landscape at Disease-Causing Polymorphic Repeat Loci. PhD Thesis, University of Toronto, Canada.

Mirkin,S.M. (2007) Expandable DNA repeats and human disease. *Nature*, **447**, 932–940.

Mudunuri,S.B. and Nagarajaram,H.A. (2007) Imex: imperfect microsatellite extractor. *Bioinformatics*, **23**, 1181–1187.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol*., **48**, 443–453.

Ohadi,M. *et al*. (2012) Evolutionary trend of exceptionally long human core promoter short tandem repeats. *Gene*, **507**, 61–67.

Parisi,V. *et al*. (2003) String: finding tandem repeats in DNA sequences. *Bioinformatics*, **19**, 1733–1738.

Pellegrini,M. *et al*. (2010) Trstalker: an efficient heuristic for finding fuzzy tandem repeats. *Bioinformatics*, **26**, i358–i366.

Pellegrini,M. *et al*. (2012) Tandem repeats discovery service (treads) applied to finding novel cis-acting factors in repeat expansion diseases. *BMC Bioinformatics*, **13**, S3.

Pemberton,T.J. *et al*. (2009) Sequence determinants of human microsatellite variability. *BMC Genomics*, **10**, 612.

Pokrzywa,R. and Polanski,A. (2010) Bwtrs: a tool for searching for tandem repeats in DNA sequences based on the burrows–wheeler transform. *Genomics*, **96**, 316–321.

Pop,P.G. (2015) DNA repeats detection using a dedicated dot-plot analysis. In: *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*. pp. 1–4. IEEE.

Rosenberg,N.A. *et al*. (2005) Clines, clusters, and the effect of study design on the inference of human population structure. *PLoS Genet*., **1**, e70.

Ruitberg,C.M. *et al*. (2001) Strbase: a short tandem repeat DNA database for the human identity testing community. *Nucleic Acids Res*., **29**, 320–322.

Saha,S. *et al*. (2008) Empirical comparison of ab initio repeat finding programs. *Nucleic Acids Res*., **36**, 2284–2294.

Sato,N. *et al*. (2009) Spinocerebellar ataxia type 31 is associated with "inserted" penta-nucleotide repeats containing (tggaa)n. *Am. J. Hum. Genet*., **85**, 544–557.

Sawaya,S. *et al*. (2013) Microsatellite tandem repeats are abundant in human promoters and are associated with regulatory elements. *PLoS One*, **8**, e54710.

Smit,A. *et al*. (2017) Repeat Masker Website.

Sokol,D. *et al*. (2007) Tandem repeats over the edit distance. *Bioinformatics*, **23**, e30–e35.

Sonnhammer,E.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.

Taneda,A. (2004) Adplot: detection and visualization of repetitive patterns in complete genomes. *Bioinformatics*, **20**, 701–708.

Thiel,T. *et al*. (2003) Exploiting est databases for the development and characterization of gene-derived ssr-markers in barley (hordeum vulgare l.). *Theor. Appl. Genet*., **106**, 411–422.

Todd,P.K. *et al*. (2013) CGG repeat-associated translation mediates neurodegeneration in fragile x tremor ataxia syndrome. *Neuron*, **78**, 440–455.

Tóth,G. *et al*. (2000) Microsatellites in different eukaryotic genomes: survey and analysis. *Genome Res*., **10**, 967–981.

Viguera,E. *et al*. (2001) Replication slippage involves DNA polymerase pausing and dissociation. *EMBO J*., **20**, 2587–2595.

Vinces,M.D. *et al*. (2009) Unstable tandem repeats in promoters confer transcriptional evolvability. *Science*, **324**, 1213–1216.

Wexler,Y. *et al*. (2005) Finding approximate tandem repeats in genomic sequences. *J. Comput. Biol*., **12**, 928–942.

Wieben,E.D. *et al*. (2012) A common trinucleotide repeat expansion within the transcription factor 4 (TCF4, E2-2) gene predicts Fuchs corneal dystrophy. *PLoS One*, **7**, e49083.

Willems,T. *et al*. (2017) Genome-wide profiling of heritable and de novo str variations. *Nat. Methods*, **14**, 590–592.

Winnepenninckx,B. *et al*. (2007) Cgg-repeat expansion in the DIP2B gene is associated with the fragile site FRA12A on chromosome 12q13.1. *Am. J. Hum. Genet*., **80**, 221–231.

Wirawan,A. *et al*. (2010) Inverter: integrated variable number tandem repeat finder. In: *Computational Systems-Biology and Bioinformatics*. Springer, Berlin, Heidelberg, pp. 151–164.

Zhou,H. *et al*. (2009) Detection of tandem repeats in DNA sequences based on parametric spectral estimation. *IEEE Trans. Inf. Technol. Biomed*., **13**, 747–755.