Forensic Science International: Genetics 30 (2017) 18–23

Contents lists available at ScienceDirect

Forensic Science International: Genetics

journal homepage: www.elsevier.com/locate/fsig

Short communication

Fast STR allele identification with STRait Razor 3.0

August E. Woerner[a,*], Jonathan L. King[a], Bruce Budowle[a,b]

[a] Center for Human Identification, University of North Texas Health Science Center, 3500 Camp Bowie Blvd., Fort Worth, TX 76107, USA
[b] Center of Excellence in Genomic Medicine (CEGMR), King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 5 April 2017
Received in revised form 8 May 2017
Accepted 29 May 2017
Available online 1 June 2017

Keywords:
STRait Razor
Short tandem repeats
Bioinformatics
Massively parallel sequencing
Flanking variation

ABSTRACT

The short tandem repeat allele identification tool (STRait Razor), a program used to characterize the haplotypes of short tandem repeats (STRs) in massively parallel sequencing (MPS) data, was redesigned. STRait Razor v3.0 performs ~660× faster allele identification than its previous version (v2s), a speedup that is largely due to a novel indexing strategy used to perform "fuzzy" (approximate) string matching of anchor sequences. Written in a portable compiled language, C++, STRait Razor v3.0 functions on all major operating systems including Microsoft Windows, and it has cross-platform multithreading support. *In silico* estimates of precision and accuracy of STRait Razor v3.0 were 100% in this evaluation and results were highly concordant with those of Strait Razor v2s. STRait Razor v3.0 adds several key features that simplify the haplotype reporting process, including simple filters to remove low frequency haplotypes as well as merging haplotypes within a locus encoded on opposite strands of the DNA molecule.

1. Introduction

STRait Razor is a bioinformatics suite used to identify and characterize sequence and length-based polymorphisms in MPS data. STRait Razor consists of two major components: The STRait Razor perl script, which identifies putative haplotypes, and the Strait Razor Excel workbook, which is used to collate, annotate and visualize said haplotypes. A complete redesign was performed on the former; a tool that until now was restricted to unix-type environments. This update provides both a stable code-base that operates on all major operating systems including Microsoft Windows and an indexing strategy tailored to the identification of sequence variants based on anchor sequences.

STRait Razor adopts a (now) relatively common approach (see Refs. [1–7]) to extracting sequence variants at known loci. In brief, a locus is first identified, and then 5′ and 3′ anchoring sequences that are near (though not necessarily adjacent to) the locus are selected. Note that STRait Razor was designed initially for capturing short tandem repeats (STRs), but it can now detect single nucleotide polymorphisms (SNPs) and insertions/deletions (indels). The presence of these anchors in the correct orientation, as well as a user-defined motif-type residing between them (e.g., CATA) is used to associate the intervening haplotype with a given locus. This haplotype then can be reduced to a length-based polymorphism call (e.g., 19.2), and the haplotype itself can be used for downstream inferences.

This STRait Razor update uses a novel indexing strategy to locate anchor sequences. Before the mechanics of this index are described some nomenclature must be described. The term *string* refers to a contiguous sequence of characters (e.g., nucleotides), with the *prefix* of length *n* of a string being its first *n* characters, and the like *suffix* of a string being its last *n* characters. A *substring* of a string then can be taken as a prefix of a suffix (or a suffix of a prefix) of a string. In general, a substring search can be exact, i.e., permitting no discordant bases, or approximate, i.e., permitting some number of mismatches between strings or substrings. String search can be further categorized into two types: offline and online (for review see [8]). For the latter, the substrings sought (e.g., the anchors) first can be organized, or indexed, with this investment in computation time ideally leading to fast search times. With the former, no index is created and the patterns are searched "as is". Indexing is a common search technique. For example, BWA [9] and bowtie2 [10] both use online strategies, organizing the reference genome into a Burrows Wheeler-transformed [11] suffix array [12] prior to approximate string search. Algorithms that identify STR-based polymorphisms have primarily used offline strategies [3–6], often relying on the bitap algorithm in the unix utility agrep [13] or tre-agrep [14], or on semi-global alignment [5] (but see [2,15], which do a mixture of techniques).

In this paper STRait Razor v3.0 is introduced, which uses a simple and effective purely online indexing strategy, organizing anchor sequences into a trie [16] (Fig. 1) to facilitate fast

* Corresponding Author.
E-mail address: August.Woerner@unthsc.edu (A.E. Woerner).

http://dx.doi.org/10.1016/j.fsigen.2017.05.008
1872-4973/

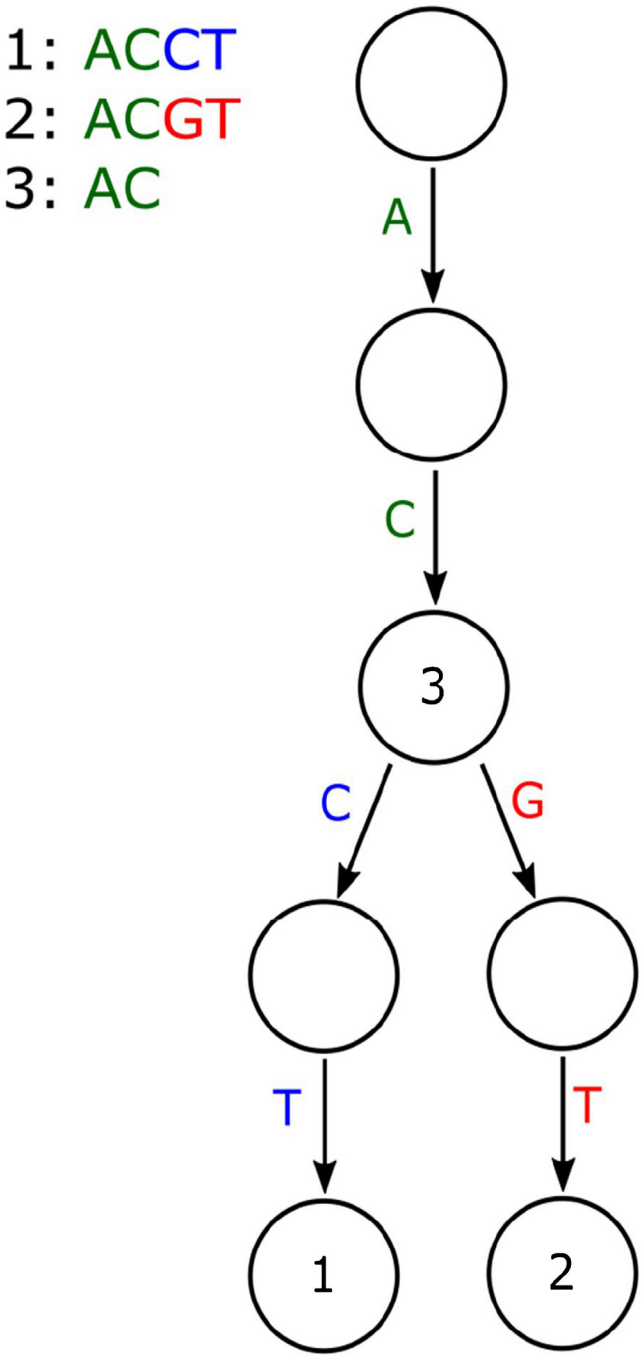

Fig. 1. A trie composed over three sequences. Each circle corresponds to a node in the trie, while labeled edges (arrows) indicate valid paths (i.e., prefixes of one or more sequences in the trie). If a node corresponds to the last letter of a word in the trie, it is marked with the index of that sequence (e.g., 3 for read AC). Shared prefixes are shown in green, while unique suffixes are shown in blue/red. Exact search of a word in a trie terminates whenever a node with the appropriate edge label fails to be found in the search, in which case all indexes found in the traversal thus far are returned. For instance, search of the word ACT would search the first 3 nodes (2 edges) as the edges A followed by C are found. The search then would record index 3, indicating that the third sequence in the trie matches a prefix of the queried word, and terminate as no edge T is found from this node. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

approximate string search. Rather than implicitly searching this trie (e.g., [9]), STRait Razor explicitly queries the trie in a manner that supports fast fuzzy substring search of MPS data. To assess the accuracy and the overall performance of STRait Razor v3.0, both *in silico* and empirical experiments were performed to assess STR variation in MPS data. *In silico* experiments were used to assess the sensitivity and accuracy of STRait Razor, while empirical experiments assessed timing and consistency of STRait Razor v3.0 with that of v2s [7].

2. Materials and methods

2.1. Algorithm description and implementation

STRait Razor applies a series of algorithms to aid in the analysis of both length-based as well as sequence-based markers in MPS data. While the previous versions of STRait Razor implemented fuzzy matching with the unix utility tre-agrep [13] and was written in a manner tailored to unix/linux operating systems, this original design strategy had practical and algorithmic limitations. In practice, the reliance on unix restricts the computational environment of STRait Razor, precluding its use in typical Windows systems. Algorithmically, previous iterations of STRait Razor applied the bitap algorithm to each anchor sequence independently. Thus, given a set of reads composed of n bases in total and k anchor sequences each of length at most m, the run-time of the original STRait Razor is $O(knm)$ (treating the Hamming distance as constant). Further, traditional implementations of bitap assume that m is at most the size of a machine word (typically 32- or 64-bits/bases), an assumption that while generally true may not always hold.

To perform faster search of anchors, STRait Razor v3.0 first organizes them into a data structure that facilitates fast substring search. While anchors can be indexed and searched using traditional alignment strategies, there are several key properties that would fail to be exploited with this design. First, the number of anchoring sequences in practice is vastly outnumbered by the number of bases searched ($k << n$), and anchor sequences themselves are typically quite short ($m \sim 30$ bp). Second, while anchor sequences are found typically using fuzzy searching, the level of approximation employed in this search is quite small (i.e., in general matching permits at most one base difference). Given that there are $3m$ possible single-base substitutions to an anchor sequence of length m, one can enumerate all permutations of all anchors explicitly in $O(km)$ time (with both k and m being small in practice).

Rather than using traditional approximate string matching techniques, STRait Razor v3.0 performs fuzzy string matching using exact string matching techniques taken over all permutations of anchor sequences. In particular, STRait Razor v3.0 first finds substrings of MPS reads that have a maximum Hamming distance of one to a given anchor (i.e., permitting at most one single-base substitution between the anchor and the substring). To do this, STRait Razor v3.0 uses a trie [16] composed of all k anchor sequences (Fig. 1) and all possible single-base substitutions of these anchors. To find matching anchors, one need only perform a top-down search of the trie, beginning initially with the first base in the read. Such a search encounters at most $m + 1$ nodes in the trie (and by proxy, only the first $O(m)$ bases in the read), and such a search finds every single anchor that is a prefix of the read permitting a single substitution (i.e., 1-base change). Search of the entire read continues by considering every possible suffix of the read; i.e., starting at the second base, third base, etc., until the read is exhausted. Thus, this trie search strategy finds all anchor sequences that match prefixes of suffixes (i.e., substrings) of a given read in a manner consistent with semi-global alignment.

This search strategy has several interesting properties. While traditional offline approaches would search for each anchor sequence consecutively, this search-style finds all anchors

simultaneously. Further, top-down trie search is, in the worst-case, invariant to both k and the number of permutations chosen over anchor sequences. Thus, each base in the read corresponds to an O(m) time operation on the trie, which implies an overall search time that is O(nm) for finding matching anchor sequences. Search times in practice are likely $\sim n$ on average, as most searches do not terminate in a matching anchor sequence and instead terminate much earlier in the trie traversal. As k, m, and the level of approximation used in search are small in practice, the memory footprint of this trie index is small. In particular, a trie composed of a single (non-permuted) anchor sequence of length m requires $m+1$ nodes in the trie, and there are $3m$ possible permutations of any anchor. Thus, to store every single base substitution of all k anchors requires O(km^2) space.

Given the above description for finding matching anchors, the algorithm of STRait Razor v3.0 on a single read is as follows:

1. For a read of length j, search the trie starting at position 1, 2, ... j in the read for matching anchor sequences.
   - For each matching anchor found, record the position and orientation of the matching anchor.
   - For each matching motif found, mark an STR as being valid if it occurs after its corresponding anchor.
2. For every STR queried:
   - If the correct anchors are found in the correct orientation, record the intervening haplotype

STRait Razor v3.0 indexes both the leading anchor (5′ of the locus), the trailing anchor (3′ of the locus), as well as their reverse complements. Matches found in line i of the above algorithm thus correspond to anchors that may or may not be in the correct orientation with respect to strand. This potential inconsistency is addressed in line *2*, which requires that both anchors be found in the same orientation, and if, say, the read is on the positive strand, then the 5′ anchor must be found first followed by the 3′ anchor. Motifs also are stored in the trie, and a STR is marked as being valid if the motif occurs after (that is, 3′ of the 5′ anchor and 5′ of the 3′ anchor) a specified anchor. The intervening haplotypes then are recorded if both the 5′ and 3′ anchors are found in their correct orientation, and they are marked as valid. The number of times each haplotype is found and the STR name and allele call are reported by STRait Razor v3.0.

STRait Razor v3.0 adds several other new features to simplify the reporting process. STRait Razor reports unique haplotypes as found on both the positive and negative strands, as they are often found natively in MPS data. As recording the same haplotype on both strands is redundant, STRait Razor v3.0 can optionally reverse complement haplotypes encoded on the negative strand, placing them onto the positive strand (under the assumption that the anchors themselves are encoded on the positive strand), as well as reporting the number of occurrences of the haplotype on both strands. STRait Razor v3.0 can also optionally filter loci by "type" (see *Locus specifications*), allowing the user to only select, say, autosomal or X-linked loci, or any other user-defined type. Low-frequency haplotypes can be removed from the output of STRait Razor v3.0, thus allowing for quick and easy thresholds to be set prior to the results being reported.

While previous versions of STRait Razor relied on haplotype lengths, and their associated allele calls, that have been defined *a priori*, STRait Razor v3.0 instead computes allele calls without expectations on length. In particular, previous versions of STRait Razor used a lookup table to associate a haplotype length with an allele length. A byproduct of this approach was that haplotypes whose lengths were not included in this table were omitted from the reporting process. STRait Razor v3.0 instead uses the more general approach taken by capillary electrophoretic (CE) methods to name haplotypes. In particular, v3.0 uses the number of canonical non-STR associated bases within the target region, termed the offset, and the canonical motif length, as they are defined in the locus configuration file. Allele lengths are then computed as the integer division of $\frac{haplotype\ \ length-offset}{motif\ \ length}$ and with the microvariant nomenclature being the remainder of this computation. This approach does not necessarily reflect the true length of the STR *per se*, but instead provides a unique name for a given haplotype length that generally corresponds to the number of repeating units in the STR as would be measured using the CE. Take, for instance, the locus D13S317 which contains a four nucleotide deletion in its flanking region. While the number of repeats within the repeat region may be 12, the CE-based allele may be an 11.

STRait Razor v3.0 is written in C++ and thus operates on all major operating systems including Windows and any of a variety of unix-based operating systems including OS X. STRait Razor v3.0 is multithreaded, allowing it to exploit multi-core computer architectures. Multithreading is implemented by statically assigning different threads to evaluate different reads in parallel, with a separate "producer" thread that loads reads into a double-buffer. STRait Razor v3.0 supports uncompressed fastq files and the standard input stream, allowing the tool to accommodate compressed file formats (e.g., bam, gzip, bzip2) and to function in analysis pipelines. STRait Razor v3.0 is both freely available and open source under the MIT license.

2.2. Locus specifications

STRait Razor v3.0 targets loci that have been identified *a priori*, and the rules for detecting these loci are described in the locus configuration file. The configuration file specifies the name of the STR, a user-defined type (e.g., Autosomal, X-linked or any other user-defined name), the 5′ and 3′ anchor sequences, and the period and offset of the STR. The anchor sequences given are assumed to be on the positive strand. STRait Razor v3.0 also allows for a quantification of the number of times a particular anchor sequence can be found. This capability allows for identification and extraction of haplotypes from loci that have been duplicated,

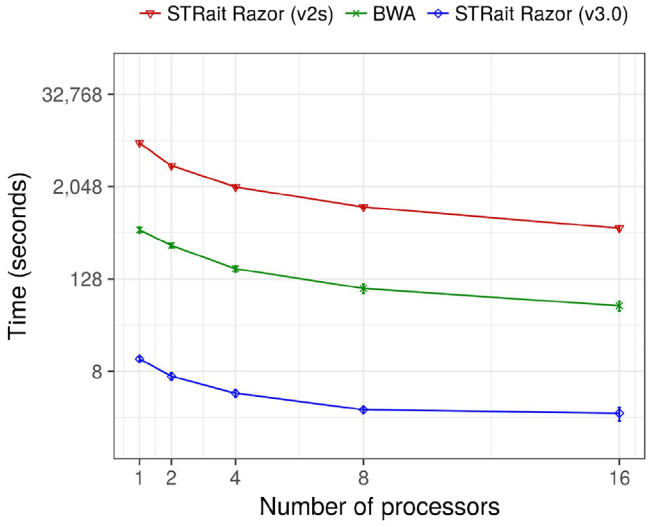

Fig. 2. Runtimes (y-axis) of STRait Razor v2s, STRait Razor v3.0 and BWA on a single sample on multiple cores (x-axis). v3.0 (diamonds) uses a trie to index anchor sequences, while v2s (triangles) uses the bitap algorithm. Runtimes for BWA mem (astrisks; mapping to the hg38 genome) are provided as a reference of typical genomic processing times. The runtimes shown are means ± 2 standard deviations estimated over ten replicates. Note the $\log_2$ scale of the y-axis.

and loci in which the anchor sequence itself also has been duplicated. One example of this function is seen with the loci DYS389I and DYS389II. As has been reported previously [17,18], both loci have identical anchor sequences, and DYS389II properly contains DYS389I. This case can be handled by specifying that the anchor sequence for DYS389II must occur twice, with STRait Razor v3.0 reporting the longer of the two possible interpretations. A byproduct of this rule is that reads where PCR artifacts have caused duplication of the anchor sequence (perhaps because of hairpins) may cause a single read to be associated with two possible intervening (and overlapping) haplotypes. In this case, STRait Razor v3.0 would report neither haplotype as the correct number of anchor sequences were not found and an unambiguous haplotype cannot be determined.

2.3. Samples, library preparation and sequencing

STRait Razor v3.0 was evaluated both with *in silico* reads (below) and with reads empirically generated. For the latter, a subset of samples from [19] were chosen to assess the speed and relative accuracy of STRait Razor v3.0. Note that the ForenSeq™ panel actually encodes an additional locus (DYS461), and as such, despite the title of [19], that study encodes a total of 59 STRs (plus Amelogenin). Briefly, anonymized samples, as per UNTHSC IRB-approved protocols, were analyzed using the ForenSeq™ DNA Signature Prep Kit the MiSeq FGx Forensic Genomics system (Illumina). All analyses were performed on a Dell PowerEdge T620 server with dual 10-core Intel Xeon E5-2670v2 2.5 GHz processors, and all plotting and statistics were performed in the statistical computing language R [20] using the graphics library ggplot2 [21].

Table 1
The number of extracted sequences from STRait Razor v3.0 (*c* v3.0) versus STRait Razor v2s (*c* v2s), as well as the haplotype differences (*d*) and atypical read counts (*a*). A single sample (C96) was run on both STRait Razor v3.0 and STRait Razor v2s, and the number of extracted sequences was computed and the haplotype identities were compared. The count (*c*) of the number of reads associated with each STR are shown for both STRait Razor v3.0 and STRait Razor v2s. Haplotype differences (*d*) are a count of the number of reads that are associated with a STR found by one program and not the other. For example *d* v2s counts the number of reads associated with a given STR that were identified by v2s and not v3.0. Note that a small absolute difference (residual) in *c* for a locus tends to imply small *d*. *a* counts the number of reads associated with an STR that were found by v3.0 and not reported by v2s, and the haplotype lengths did not match the *a priori* expectations of v2s. Loci with an absolute residual in $c < 2$ are omitted.

Locus	c (v2s)	c (v3.0)	*d* (v2s)	*d* (v3.0)	a
DYF387S1	3014	3135	130	9	3
DYS389II	2024	2123	124	25	0
DYS389I	12,135	12,071	54	118	118
DYS19	1181	1231	50	0	0
vWA	1901	1938	37	0	0
DYS448	3308	3287	41	62	18
DYS533	2565	2582	17	0	0
D1S1656	1752	1762	10	0	0
D8S1179	7682	7691	9	0	0
D5S818	1794	1803	9	0	0
D18S51	3792	3799	7	0	0
D6S1043	7649	7654	5	0	0
DYS385	6915	6910	0	5	5
DYS461	1085	1089	5	1	1
DYS576	10,003	10,007	5	1	1
TPOX	4696	4700	4	0	0
DXS8378	4025	4029	4	0	0
DYS481	2778	2776	0	2	2
D21S11	2267	2269	3	1	1
CSF1PO	2160	2162	2	0	0
Amelogenin	3166	3164	1	3	3

2.4. In silico reads

To assess the sensitivity of STRait Razor v3.0, locus configurations from [19] were used to generate reads that *a priori* are associated with a given STR marker. The configuration file of Strait Razor v2s [7] specifies the two anchor sequences, the STR motif pattern, and valid (i.e., known) lengths between the anchors and their associated length-based STR nomenclature. Using this information, synthetic reads parameterized off of the 59 markers (plus Amelogenin) in [19] were generated using the following algorithm:

1. For a given marker a strand, positive or negative, was chosen at random.
2. Then, anchor sequences a_1 and a_2, and the motif m associated with that strand were selected.
3. One random base was selected within both a_1 and a_2, changing these bases to a random nucleotide. Note that there is a $\sim 1/4$ probability of the base remaining the same with this operation.
4. Using the marker lengths from the configuration file from [19], a known distance between a_1 and a_2 was selected at random, and a string S of nucleotides of that length were generated. Then the motif m was overlaid onto S.
5. Last, two strings, s_1 and s_2, of nucleotides of length [0,50] were generated and used the concatenation of: s_1, a_1, S, a_2, s_2 as the DNA sequence of our *in silico* read.

Random herein means a uniform distribution as generated by the Perl function rand. To avoid the possibility of a read being erroneously associated with the wrong STR, strings S, s_1, and s_2 were formed out of the least-frequent nucleotide in the configuration file ("G"). This algorithm guaranteed that a single *in silico* read would be associated with exactly one STR, simplifying interpretation of the result. Of note, due to a tandem duplication, the 5′ anchor for the DYS389II locus is expected to be found twice. Thus, the *in silico* strategy was extended to include an additional anchor (a_1) and random string (s_1) for this locus. STRait Razor v3.0 was run on each read individually, thus assessing if the correct STR and only the correct STR is found for a given read.

2.5. Timing, concordance and bias experiments

STRait Razor v3.0 was benchmarked against STRait Razor v2s using the default maximum Hamming distance of 1 for both programs. Benchmarks were performed on the same 59 STRs (plus Amelogenin) described in [19]. STRait Razor v2s runtimes were conducted solely on the runtimes of the perl script, and not of the Excel workbook. For a reference, BWA [9] also was used to map reads to the hg38 genome using the mem mapping algorithm set to the default parameters. Execution times were measured using the bash built-in utility time, with the mean and standard deviation run-times computed over 10 independent replicates. Run-times and concordance were compared for a single sample (C96) with the highest read-depth (530,144 351 bp reads) amongst the samples from [19].

Concordance was further investigated with an additional 9 arbitrarily chosen individuals from [19] and was assessed using three summary statistics. The first statistic, *c*, simply counts the number of reads associated with a given haplotype in STRait Razor v3.0 and STRait Razor v2s. The second statistic, *d*, counts the number of reads that were associated with a STR for one version of STRait Razor and not the other. For example, if a locus has haplotypes *X*, Y, and *Z*, and v2s found each haplotype 5, 3, and 1 times and v3.0 found each haplotype 7, 4 and 0 times, the corresponding *d* values are 3 (2 + 1 + 0) and 1 (0 + 0 + 1) for v3.0 and v2s, respectively. The third summary statistic, *a*, counts the

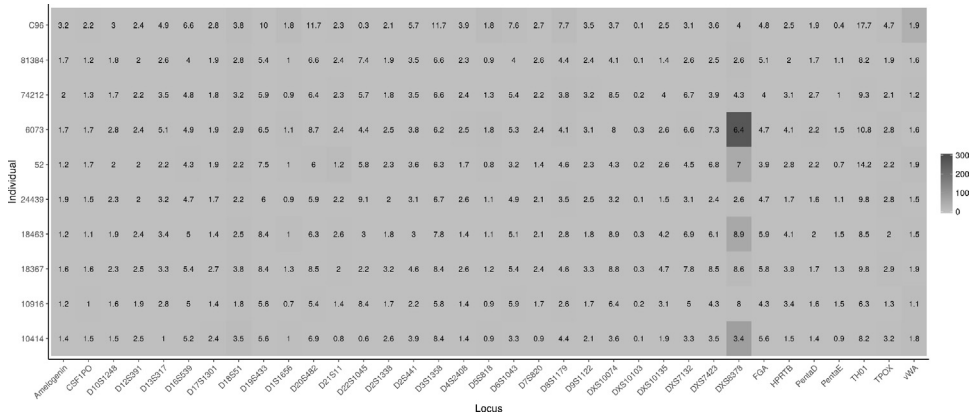

Fig. 3. Heatmap of locus coverage and haplotype differences in 10 individuals across the autosomes and X chromosome. We tabulated the total number of reads associated with a given locus (columns, using STRait Razor v3.0) across individuals (rows), with that count (in 1000s) shown in text. The difference in haplotype counts (*d*) estimated by STRait Razor v2s vs STRait Razor v3.0, estimated solely in the case where v2s found more of a given haplotype than STRait Razor v3.0, is shown in greyscale.

number of "atypical" haplotype lengths found by v3.0. In particular, *a* is the number of reads whose corresponding haplotypes were found by v3.0 and whose haplotype length was not encoded in the configuration file of [19] and thus were excluded from the reporting process of v2s.

Consistency was estimated with respect to read mapping with BWA. In particular, reads were mapped to the HG38 reference genome using BWA-MEM, and samtools view and samtools flagstat were used to extract reads that overlapped the loci. The accuracy of STRait Razor v2s has been reported previously [3,6], and as such the concordance of STRait Razor v3.0 with STRait Razor v2s serves as an additional assessment of accuracy.

3. Results

3.1. In silico precision

For the *in silico* experiment 6 million synthetic reads were generated; 100,000 reads for each of the 60 markers (includes DYS461). Each read was run through STRait Razor v3.0 individually, and the associated STR was recorded. STRait Razor v3.0 reported each of the 100,000 records correctly for every STR, implying 100% precision and accuracy for these loci. Running all 6 million reads collectively yielded the correct number of STRs recovered for all STRs, and this result held when STRait Razor v3.0 was run with and without multi-threading. Taken together, STRait Razor v3.0 does not appear to miss any loci when they conform to the specifications detailed in the configuration file, and the results indicate that there are no serious bugs in the new software version.

3.2. Processing times

As seen in Fig. 2 (note log scale), STRait Razor v3.0 is considerably faster than its predecessor, with single-core processing times decreasing by ~660×, going from an average of ~2 h (7627 s) to 11 s for a single high-coverage sample. For a reference, read mapping with BWA took ~9 min (557 s) on average on a single core. Both STRait Razor v3.0 and STRait Razor v2s employ multicore processing techniques, and as measured by the slopes in Fig. 2, both versions appear to scale similarly though there may be more speed gains with increasing core count for the version of STRait Razor v2s and with BWA than for STRait Razor v3.0. These speed gains may be a byproduct of reading large fastq files, an IO intensive activity that places a lower bound on the (best-case) overall processing time of STRait Razor v3.0.

3.3. STRait Concordance

The consistency of STRait Razor v3.0 was compared to that of STRait Razor v2s for sample C96. Of the 530,144 reads evaluated, the sequence of 252,139 and 252,470 STRs were inferred by STRait Razor v3.0 and Strait Razor v2s, respectively. Read mapping with BWA yielded 250,218 reads which overlapped the loci, suggesting that ~50% of read-recovery is expected with the ForenSeq™ panel. This difference likely reflects the additional SNPs also targeted by this panel. The number of reads associated with each STR were highly consistent between two versions of STRait Razor, yielding a Pearson correlation coefficient >99.99% (see also Table 1).

To test if the haplotype identities remained the same between STRait Razor v2s and v3.0 the *c*, *d* and *a* statistics were computed in sample C96 (Table 1), and in C96 and 9 other individuals (Fig. 3) on the autosomes and the X chromosome. Some differences are expected between these programs; while STRait Razor v2s had no explicit rules for tie-breaking when multiple matching anchor sequences are found, STRait Razor v3.0 has a conservative set of rules for extracting sequences. Namely, in STRait Razor v3.0 sequences are extracted only if they can unambiguously be associated with a STR, while with STRait Razor v2s if multiple matches are found then ties are broken arbitrarily. Further, STRait Razor v3.0 allows for the inference of the number of repeats in the STR directly based on the sequence length, while v2s required that the sequence length match *a priori* expectations. Taken together, STRait Razor v3.0 may report less reads at a locus because of its tie-breaking rules, or it may report more loci because of its lack of *a priori* expectations on read lengths.

The *a* statistic computes the number of alleles of "atypical" length, and outside of the *a priori* expectations employed by [19], parsing out some of these reporting differences. The per-locus *a* statistic across the autosomes and X chromosome is <5 in all individuals, which suggests that these *a priori* expectations likely have little impact on the downstream analyses of [19]. However, even if not apparent in this study, *a priori* assumptions on allele size have the potential to induce allelic dropout, an issue that may become relevant in populations with small effective population size (e.g., isolates), with a large degree of population structure, or if tested on extremely large sample sizes.

In practice, most of the differences seen between v2s and v3.0 are clustered in the duplicated loci DYS389I, DYS389II, and DYF387S1 (Table 1). For the DYS389II locus, STRait Razor v3.0 requires the 5′ anchor to occur exactly twice in the sequence; if it is present only once, as may become the case if the sequence is sufficiently degraded, it becomes consistent with the DYS389I

locus. The locus DYF387S1, on the other hand, has a 3′ anchor that contains low-complexity sequence (its prefix is $A_4T_1A_7$), which increases the probability of a second anchor match occurring simply by chance and thus slightly reducing the frequency of this marker in STRait Razor v3.0. With respect to haplotype differences (d), the vast majority of loci show little to no differences in the haplotype counts (grayscale, Fig. 3), with only 8 of the 350 cells in Fig. 3 having a $d > 9$. Further, even when differences are found, they are small with respect to the overall coverage for a locus (text, Fig. 3). The one exception may be the locus DXS8378, which appears to show more haplotypes being omitted by STRait Razor v3.0 than by v2s, especially in sample 6073. Manual inspection of this locus in this sample shows that the 3′ anchor is apparently repeated in some reads; as stated above, this potential duplication would cause STRait Razor v3.0 to drop the haplotype as the haplotype identity becomes ambiguous, while in this case STRait Razor v2s chooses the shorter of the two alleles. Overall, very few haplotypes are missed by v2s and found by STRait Razor v3.0, with a maximum of 4 seen across autosomal and X-linked loci in these samples (data not shown).

It should be noted that the allowing a maximum Hamming distance of one may lead to allele dropout in a sample containing multiple mismatches in an anchor. When converting configuration files from v2s to v3.0, two anchors were found to contain phased SNPs in at least one individual (n = 2504) from the 1000 Genome Project (Phase 3) [22]. While the configuration files have been modified in both v2s and v3.0 to account for these loci, other low-frequency variants may exist in the greater population(s) and our and other software may experience at some level bioinformatic "dropout".

5. Conclusions

STRait Razor v3.0 provides a marked improvement of the allele identification strategy employed by its previous versions. STRait Razor v3.0 is fast, in part owing to its development in a compiled programming language (C++) and in part to an indexing strategy that is tailored for fast approximate search of anchor sequences. STRait Razor v3.0 is multithreaded, allowing it to exploit multicore CPUs. Further, it no longer requires regular files and instead can use the standard input stream, allowing it to process compressed file formats and generally function in analysis pipelines. STRait Razor v3.0 also has the ability to selectively filter loci by type (e.g., one can attempt to find only autosomal loci or any other user-defined type). The tool can optionally place haplotypes onto the positive strand, and low-frequency haplotypes can be excluded from the output, thus simplifying the reporting process. STRait Razor v3.0 also uses a conservative set of requirements used to resolve ambiguous haplotype identities, and it has the ability to compute the length-based nomenclature of STRs without prior expectations on haplotype lengths.

STRait Razor v3.0 is open source and freely available at https://github.com/Ahhgust/STRaitRazor and at https://www.unthsc.edu/graduate-school-of-biomedical-sciences/molecular-and-medical-genetics/laboratory-faculty-and-staff/strait-razor, along with standard configuration files that will enable it to be used on a variety of MPS systems. Comments and findings that could improve the performance of STRait Razor v3.0 are welcome and encouraged.

Funding

This work was supported in part by award no. 2015-DN-BX-K067, awarded by the National Institute of Justice, Office of Justice Programs, U.S. Department of Justice.

Conflict of interest

None.

Acknowledgements

The opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect those of the U.S. Department of Justice. The authors would like to thank the two anonymous reviewers for their feedback and suggestions to earlier versions of this manuscript.

References

[1] S.L. Fordyce, M.C. Ávila-Arcos, E. Rockenbauer, C. Børsting, R. Frank-Hansen, F.T. Petersen, E. Willerslev, A.J. Hansen, N. Morling, T.P. Gilbert, High-throughput sequencing of core STR loci for forensic genetic investigations using the Roche Genome Sequencer FLX platform, Biotechniques 51 (2011) 127–133, doi:http://dx.doi.org/10.2144/000113721.
[2] M. Gymrek, D. Golan, S. Rosset, Y. Erlich, lobSTR: A short tandem repeat profiler for personal genomes lobSTR: A short tandem repeat profiler for personal genomes, Genome Res. (2012) 1154–1162, doi:http://dx.doi.org/10.1101/gr.135780.111.
[3] D.H. Warshauer, D. Lin, K. Hari, R. Jain, C. Davis, B. Larue, J.L. King, B. Budowle, STRait Razor A length-based forensic STR allele-calling tool for use with second generation sequencing data, Forensic Sci. Int. Genet. 7 (2013) 409–417, doi:http://dx.doi.org/10.1016/j.fsigen.2013.04.005.
[4] S.L. Friis, A. Buchard, E. Rockenbauer, C. Børsting, N. Morling, Introduction of the Python script STRinNGS for analysis of STR regions in FASTQ or BAM files and expansion of the Danish STR sequence database to 11 STRs, Forensic Sci. Int. Genet. 21 (2016) 68–75, doi:http://dx.doi.org/10.1016/j.fsigen.2015.12.006.
[5] S.Y. Anvar, K.J. Van Der Gaag, J.W.F. Van Der Heijden, Veltrop M.H.A.M, Vossen R.H.A.M, R.H. De Leeuw, C. Breukel, H.P.J. Buermans, J.S. Verbeek, P. De Knijff, J. T. Den Dunnen, J.F.J. Laros, TSSV: A tool for characterization of complex allelic variants in pure and mixed genomes, Bioinformatics 30 (2014) 1651–1659, doi:http://dx.doi.org/10.1093/bioinformatics/btu068.
[6] D.H. Warshauer, J.L. King, B. Budowle, STRait razor v2.0: the improved STR allele identification tool-Razor, Forensic Sci. Int. Genet. 14 (2015) 182–186, doi:http://dx.doi.org/10.1016/j.fsigen.2014.10.011.
[7] J.L. King, F.R. Wendt, J. Sun, B. Budowle, STRait Razor v2s: Advancing sequence-based STR allele reporting and beyond to other marker systems, Forensic Sci. Int. Genet. 29 (2017) 21–28, doi:http://dx.doi.org/10.1016/j.fsigen.2017.03.013.
[8] G. Navarro, A guided tour to approximate string matching 1 introduction, ACM, Comput. Surv. 33 (2001) 31–88, doi:http://dx.doi.org/10.1145/375360.375365.
[9] H. Li, R. Durbin, Fast and accurate short read alignment with Burrows-Wheeler transform, Bioinformatics 25 (2009) 1754–1760, doi:http://dx.doi.org/10.1093/bioinformatics/btp324.
[10] B. Langmead, S.L. Salzberg, Fast gapped-read alignment with bowtie 2, Nat. Methods 9 (2012) 357–359, doi:http://dx.doi.org/10.1038/nmeth.1923.
[11] M. Burrows, D. Wheeler, A block-sorting lossless data compression algorithm, SRC Res. Rep. 124 (1994).
[12] U. Manber, G. Myers, Suffix arrays a new method for on-Line string searches, SIAM J. Comput. 22 (1993) 935–948.
[13] S. Wu, U. Mamber, Agrep – a fast approximate pattern matching tool, Proc. Winter 1992 USENIX Conf., San Fr. USA. Berkeley, 1992, pp. 153–162.
[14] V. Laurikari, TRE. The free and portable approximate regex matching library, (n.d.). http://laurikari.net/tre.
[15] G. Highnam, C. Franck, A. Martin, C. Stephens, A. Puthige, D. Mittelman, Accurate human microsatellite genotypes from high-throughput resequencing data using informed error profiles, Nucleic Acids Res. 41 (2013) 1–7, doi:http://dx.doi.org/10.1093/nar/gks981.
[16] R. De La Briandais, File Searching Using Variable Length Keys, ACM, 1959, pp. 295–298.
[17] C. Jobling, Mark A. Tyler-Smith, Fathers and sons: the Y chromosome and human evolution, Trends Genet. 11 (1995) 449–456.
[18] L. Kayser, Caglia Manfred, A. Fretwell, N. Gehrig, C. Graziosi, G. Heidorn, F. Herrmann, S. Herzog, B. Hidding, M. Honda, K. Jobling, M. Krawczak, M. Leim, K. Meuser, S. Meyer, E. Oesterreich, W. Pandya, A. Parson, W. Penacino, G. Perez-Leza, Evaluation of Y-chromosomal STRs: a multicenter study, Int. J. Legal Med. 110 (1997) 125–133.
[19] N.M.M. Novroski, J.L. King, J.D. Churchill, L.H. Seah, B. Budowle, Characterization of genetic sequence variation of 58 STR loci in four major population groups, Forensic Sci. Int. Genet. 25 (2016) 214–226, doi:http://dx.doi.org/10.1016/j.fsigen.2016.09.007.
[20] R. Core Team R: A Language and Environment for Statistical Computing, https://www.R-Project.org/. (2016). http://www.r-project.org/.
[21] H. Wickham, ggplot2: Elegant Graphics for Data Analysis, 2009. http://ggplot2.org.
[22] 1000 Genomes Project Consortium, A global reference for human genetic variation, Nature. 526 (2015) 68–74. 10.1038/nature15393.