Sequence analysis

Kmer-SSR: a fast and exhaustive SSR search algorithm

Brandon D. Pickett, Justin B. Miller and Perry G. Ridge*

Department of Biology, BYU, Provo, UT 84602, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on July 19, 2017; revised on August 16, 2017; editorial decision on August 20, 2017; accepted on August 29, 2017

Abstract

Motivation: One of the main challenges with bioinformatics software is that the size and complexity of datasets necessitate trading speed for accuracy, or completeness. To combat this problem of computational complexity, a plethora of heuristic algorithms have arisen that report a ‘good enough’ solution to biological questions. However, in instances such as Simple Sequence Repeats (SSRs), a ‘good enough’ solution may not accurately portray results in population genetics, phylogenetics and forensics, which require accurate SSRs to calculate intra- and inter-species interactions.

Results: We present Kmer-SSR, which finds all SSRs faster than most heuristic SSR identification algorithms in a parallelized, easy-to-use manner. The exhaustive Kmer-SSR option has 100% precision and 100% recall and accurately identifies every SSR of any specified length. To identify more biologically pertinent SSRs, we also developed several filters that allow users to easily view a subset of SSRs based on user input. Kmer-SSR, coupled with the filter options, accurately and intuitively identifies SSRs quickly and in a more user-friendly manner than any other SSR identification algorithm.

Availability and implementation: The source code is freely available on GitHub at https://github.com/ridgelab/Kmer-SSR.

Contact: perry.ridge@byu.edu

1 Introduction

Simple sequence repeats (SSRs) are short repetitive regions of DNA where at least one base is tandemly repeated many times due to slipped-strand mispairing and errors occurring in DNA replication, repair, or recombination (Levinson and Gutman, 1987). For decades, SSRs have been studied to determine phenotypic differences caused by increased copy numbers of short repetitive sequences (Kashi and King, 2006). Moreover, SSRs account for quantitative genetic variation and phenotypic differences without lowering species fitness (Kashi et al., 1997). SSR concentration varies not only between different species, but also between different chromosomes within the same species, and cannot be explained by assessing the nucleotide composition of sequences (Katti et al., 2001). Because SSRs reveal characteristic functions of DNA replication, recombination and repair, they are important in studying biological systems interactions, as well as studying repeat expansion-based diseases with next-generation sequencing data (Kashi and King, 2006).

Many different approaches have been used to identify SSRs. Here, we propose the use of k-mers. The term k-mer refers to a subsequence of length ‘k’ derived from a given sequence, while k-mer decomposition refers to all possible substrings of length ‘k’ that can be made from a sequence. Uses for k-mer decomposition have previously been outlined in instances such as genome assembly and machine learning (Chikhi and Medvedev, 2014; Ghandi et al., 2014). Although k-mers have been used to identify similar subsequences as in (Han et al., 2007), to our knowledge SSR identification has never been attempted through k-mer decomposition.
Kmer-SSR utilizes k-mer decomposition to provide an exhaustive or filtered approach to finding all SSRs in a given sequence (Figs 1 and 2). Our version of k-mer decomposition works by identifying all subsequences of length ‘k’ while tracking the start position of each k-mer. K-mer lengths are defined by the user as the SSR period length. Kmer-SSR minimizes the usage of random access memory (RAM) by performing k-mer decomposition and only storing k-mers that are the same as the preceding k-mer (SSR period length). If a k-mer is not identical to a k-mer found k bases previously, the previously identified k-mers will be discarded and k-mer decomposition will occur for the rest of the sequence.

2. Materials and methods

2.1 Overview

Kmer-SSR checks each position starting at the leftmost position of the sequence for each SSR period size (i.e. k-mer length) given by the user. This method allowed us to store only a single potential SSR and immediately either discard it if it was not repeated or write it to a file if it was a valid SSR.

2.2 Memory requirements

We used the following techniques to limit memory requirements:

1. Identify SSRs from left to right:
   Kmer-SSR checks each position starting at the leftmost position of the sequence for each SSR period size (i.e. k-mer length) given by the user. This method allowed us to store only a single potential SSR and immediately either discard it if it was not repeated or write it to a file if it was a valid SSR.

2. Identify SSRs with the largest period size first:
   Since Kmer-SSR does not store previously identified SSRs in memory, it is necessary to search for SSRs in a specific order, or else risk reporting SSRs fully enclosed within larger SSRs. To avoid this issue, we take the period sizes given by the user and search for SSRs from the longest period size to the smallest (e.g. if the user wants to search for 2-mers and 7-mers, we search for all 7-mer SSRs in the sequences before we search for 2-mer SSRs). When an SSR is discovered, an atomicity check is conducted to determine if the k-mer can be broken down to a smaller subsequence. An SSR is considered atomic if no smaller SSRs exist inside the first period. For example, ATATATAT would be identified as a 4-mer (ATAT) repeated twice, but ATAT is not atomic because AT (repeated twice) occurs within the first period. Thus, it is ignored because it is an invalid 4-mer and, if the user requested searching for 2-mers, it would be discovered again as a 2-mer (AT) repeated four times. If the atomicity check fails, the SSR is not reported. When an atomic (i.e. valid) SSR is discovered, the iterator moves just past the SSR, minus the current period size being searched, to ensure that overlapping SSRs are identified. For example, ACAACAACACACACAC has ACA repeated three times starting at position 0. Additionally, AC repeats five times starting at position 6. After finding the ACA repeat, we would miss the full AC repeat if we skipped to the end of the ACA repeat and resumed searching from there. Only by backtracking as described above (9–3 = 6), do we find the full AC repeat. Note that each of the nucleotides between positions 0 and 5 need not be searched for SSRs because Kmer-SSR has already found SSRs with larger period sizes than the current period size. In other words, since Kmer-SSR has already found SSRs with larger period sizes, the maximum possible overlap with the current SSR (ACA) and an adjacent following SSR is k (which is three in this example), removing the need to search for SSRs from the start of a valid SSR to k bases from the end of that SSR.

3. Create a Boolean filter array:
   To ensure that SSRs are unique and do not end in the same positions, we created a Boolean filter array of the same length as the sequence being analyzed, which is initiated to false. In C++, the implementation of this array only requires one bit per position, so the memory requirement is nominal. When an SSR is discovered, we first ensure that at least one position in the first or last SSR period size on either end of the SSR is false in the Boolean array. If one position is false, we assign all values within the array that correspond to all positions in the SSR to true. The filter allows us to ignore completely overlapping SSRs because overlapping SSRs will be set to ‘true’ at the positions of the end of the SSR.

By utilizing the above-mentioned methods, we were able to limit the amount of RAM needed to O(n), where n is the sequence length, and the constant value is slightly more than one byte (one byte to store each sequence base and one bit allocated in the Boolean filter for each base).

2.3 SSR filters

Next, we implemented a comprehensive filter that allows users to control the output of Kmer-SSR based on atomicity, cyclic duplicates, enclosed SSRs, minimum SSR length and specific SSR period sizes. Pseudocode for Kmer-SSR is in Figure 2. The following are different filters that are optionally applied to the output of Kmer-SSR:

1. Atomicity check:
   The atomicity check ensures that the smallest period size for each SSR is reported. For instance, if an ATAT repeats four

Consider a Simple Sequence Repeat (SSR), ACG, that repeats 6 times beginning at the 100th nucleotide in some sequence:

| position | 0 | 1 | 1 | 1 | 1 | 1 |
|----------|---|---|---|---|---|---|
|          | 0 | 0 | 0 | 0 | 1 | 1 |
|          | 0 | 3 | 6 | 9 | 2 | 5 |

...ACCGTTTACGACGACGACGACGACGACGGGATCCGA...

When performing k-mer decomposition, an ordered list of positions where a particular k-mer occurred is conceptually trivial to generate. In our example, k is 3. Once obtained, one can easily spot an SSR by finding runs of positions that are exactly k apart. Even if one is not doing k-mer decomposition for another purpose (e.g., sequence alignment), a simple scan of the sequence could also generate the list.

ACG: [..., 7, 53, 86, 100, 103, 106, 109, 112, 115, 207, 231, ...]
times, it would be reported as an AT repeated eight times because AT is the smallest period size within ATAT.

2. Cyclic duplicates:
Many SSRs create equally viable SSRs with slightly different positions reported. For instance, in the sequence ATATATA TATATATATA, it is arguably equally valid to report the AT repeated eight times starting at position zero as it would be to report TA repeating eight times starting at position one. To avoid duplicate reporting of cyclic duplicates and ensure the longest SSR is always reported, we choose and report only the left-most SSR. So, in this instance, only the AT repeated eight times would be reported.

3. Enclosed SSRs:
Occasionally, SSRs might be completely enclosed within other SSRs. For example, in the sequence TAAAATTAAAATT AAAAAAAT, the SSR TAAAAT is repeated three times, but within each TAAAAT there is an A that repeats four times. In this case, we only report the longest SSR, TAAAAT, repeated three times.

4. SSR length:
We allow the user to input minimum and maximum SSR lengths via command line options. By default, SSRs are only reported if they are at least 16 nucleotides long.

5. Set specific period sizes:
We allow the user to input specific period sizes to be checked (e.g., 1, 3, 5 would look for SSRs with period sizes of one, three, and five), or ranges of period sizes (e.g., 1–7 would look for SSRs with period sizes one through seven). By default, Kmer-SSR reports SSRs of period sizes one through seven. SSRs outside of the user-specified range are not reported.

6. Number of repeats:
We allow the user to input minimum and maximum numbers of repeats via command line options. By default, SSRs must repeat at least twice to be reported.

7. Enumerated SSRs:
If the user is interested in a very limited set of SSRs, they may specify those via a command line option and no other SSRs will be reported.

8. Sequence length:
The user may specify minimum and maximum bounds on the length of an input sequence, outside of which the program will not search or report SSRs. By default, if a sequence is less than 100 bases or more than 500 megabases, it will be ignored.

3 Results
We conducted pairwise comparisons of Kmer-SSR against the following SSR identification algorithms: GMATo (Wang et al., 2013), MREPS (Kolpakov et al., 2003), PRoGeRF (Lopes et al., 2015), QDD (Meglécz et al., 2014), SA-SSR (Pickett et al., 2016), SSR-
pipeline (Miller et al., 2013), SSRIT (Temnykh et al., 2001) and TRF (Benson, 1999). These comparisons were performed on DNA sequences from six different species (whole genome assembly unless otherwise noted): Anolis carolinensis chromosome 6 (CM000942.1), Chlamydomonas reinhardtii (assembly v3.5) (Merchant et al., 2007), Danio rerio chromosome 25 (CM002909.1), Dicystostelium discoideum (GCA_000004495.1), Physcomitrella patens chromosome 1 (assembly v3.3) and Saccharomyces cerevisiae (GCA_001634464.5). Table 1 displays the computational time of each algorithm and the number of SSRs correctly identified for each dataset (CPU Time and Real Time columns).

Because Kmer-SSR is multithreaded and robust to fasta files with unknown nucleotides, the real time for SSR identification using Kmer-SSR is faster than any other algorithm. Although MREPS reports a faster real time identification of SSRs, the program does not usually run with sequences containing unknown characters. With the addition of the time necessary to make the input fasta files usable for MREPS, it underperformed Kmer-SSR in all six datasets (Table 1, RealTime column). We found that with the exception of TRF, all algorithms tested were 100% accurate in identifying SSRs; however, only Kmer-SSR, MREPS and SSRIT reported all possible filtered SSRs within the range specified for each dataset (Table 1, SSRs Shared).

### Table 1. Comparisons of all nine SSR-identification algorithms across six genomes with period sizes of 1–7 and a minimum SSR length of 16 bases

| Software       | CPU Time (mm:ss) | Real Time (mm:ss) | Adjustments | Unique to Software | Unique to Kmer-SSR | Correct & Fixed | Percent Correct & Fixed |
|----------------|------------------|-------------------|-------------|--------------------|--------------------|-----------------|-------------------------|
| **Anolis carolinensis** (chr 6) |                  |                   |             |                    |                    |                 |                         |
| GMATo          | 3:28             | 2:38              | 20623008    | 16369297           | 16871              | 16870           | 100                     |
| MREPS          | 0:09             | 0:09              | 25639       | 25639              | 18284              | 18284           | 100                     |
| ProGeRF        | 18:07            | 18:07             | 16841656    | 16840821           | 17763              | 17762           | 17763                   |
| QDD            | 19:11            | 19:11             | 60994       | 60994              | 18009              | 18009           | 100                     |
| SA-SSR         | 33:47            | 33:55             | 18166       | 18166              | 18166              | 18166           | 18166                   |
| SSR-Pipeline   | 61:55            | 61:55             | 19732822    | 17301120           | 18044              | 18044           | 18044                   |
| SSRIT          | 1:29             | 1:29              | 8707        | 7412               | 18284              | 18284           | 18284                   |
| TRF            | 2:09             | 2:09              | 422851      | 411644             | 42157              | 13872           | 13707                   |
| **Chlamydomonas reinhardtii**          |                  |                   |             |                    |                    |                 |                         |
| GMATo          | 3:30             | 3:30              | 26512280    | 21624294           | 50401              | 50401           | 50401                   |
| MREPS          | 0:14             | 0:14              | 94875       | 94875              | 50752              | 50752           | 50752                   |
| ProGeRF        | 37:55            | 37:55             | 807102     | 8020213            | 32043              | 31989           | 32004                   |
| QDD            | 8:51             | 8:51              | 216943      | 216943             | 55470              | 55470           | 55470                   |
| SA-SSR         | 1324:33          | 167:48            | 56833       | 56833              | 56833              | 56833           | 56833                   |
| SSR-Pipeline   | 632:10           | 632:10            | 26973434    | 23032838           | 56729              | 56729           | 56729                   |
| SSRIT          | 2:00             | 2:00              | 310109      | 252232             | 50752              | 50752           | 50752                   |
| TRF            | 8:52             | 8:52              | 1022145     | 990316             | 181973             | 25451           | 47773                   |
| **Danio rerio** (chr 25) |                  |                   |             |                    |                    |                 |                         |
| GMATo          | 1:12             | 1:12              | 9301860    | 7537549            | 22546              | 22546           | 22546                   |
| MREPS          | 1:10             | 1:10              | 22099      | 22099              | 22099              | 22099           | 22099                   |
| ProGeRF        | 8:14             | 8:14              | 61668629    | 7690120           | 21279              | 212688         | 212688                  |
| QDD            | 7:43             | 7:43              | 49016      | 49016              | 28105              | 28105           | 28105                   |
| SA-SSR         | 2075:03          | 648:00            | 21862      | 21862              | 21862              | 21862           | 21862                   |
| SSR-Pipeline   | 1958:54          | 1598:45           | 8948450    | 7954899            | 21857              | 21857           | 21857                   |
| SSRIT          | 0:43             | 0:43              | 69645      | 58065              | 22099              | 22099           | 22099                   |
| TRF            | 5:03             | 5:03              | 293378     | 283764             | 40343              | 11255           | 16911                   |
| **Dictyostelium discoideum** |                  |                   |             |                    |                    |                 |                         |
| GMATo          | 1:02             | 1:02              | 881067     | 7126425            | 82643              | 82643           | 82526                   |
| MREPS          | 0:05             | 0:05              | 121835     | 121835             | 91681              | 91681           | 91681                   |
| ProGeRF        | 11:42            | 11:42             | 4629786    | 4604499            | 60176              | 60174           | 60174                   |
| QDD            | 3:44             | 3:44              | 171686     | 171686             | 88017              | 88017           | 88017                   |
| SA-SSR         | 723:31           | 236:01            | 90700      | 90700              | 90700              | 90700           | 90700                   |
| SSR-Pipeline   | 246:35           | 246:35            | 9292900    | 7397561            | 90810              | 90810           | 90810                   |
| SSRIT          | 0:42             | 0:42              | 265894     | 202531             | 91681              | 91681           | 91681                   |
| TRF            | 17:30            | 17:30             | 642904     | 602031             | 178902             | 40772           | 75742                   |
| **Physcomitrella patens** (chr 1) |                  |                   |             |                    |                    |                 |                         |
| GMATo          | 0:58             | 0:58              | 7981869    | 6500395            | 77397              | 77397           | 77397                   |
| MREPS          | 0:04             | 0:04              | 8787       | 8787              | 8787               | 8787           | 8787                    |
| SSR-Pipeline   | 1948:06          | 1948:06           | 7763141    | 6874175            | 8720               | 8720           | 8720                    |
| SSRIT          | 0:35             | 0:35              | 39472      | 35944              | 8787              | 8787           | 8787                    |

(Continued)
We ran all comparisons on a 2.3 GHz Intel Haswell processor. Although each algorithm was given the same amount of memory and CPUs, due to hardware variability of the CPU, runtimes could vary by up to 20%. Also, MREPS required pre-processing of the fasta files, which typically added anywhere from a few seconds to several minutes to the runtime (not depicted in the table), depending on the pre-processing approach used. Similarly, we did not include the time required to edit SSRIT and QDD’s source code in order for their programs to function over the period sizes in these tests. SSR-Pipeline could not finish searching for 1-mers in chromosome 6 of the *Anolis carolinensis* in 21 days of runtime. Accordingly, the chromosome was split into 24 approximately equal sized chunks (i.e. approximately 3.3 Mb each) and each chunk was searched for 1-mers separately by SSR-Pipeline. The required time for each chunk was summed (approximately 3 hours) and used in place of 304 hours (21 days).

The SSRS After Adjustments column reflects the number of SSRS that we did not remove or alter for purposes of making the comparison simpler. SSRS that were exact duplicates, duplicates with only the repeat number varying, duplicates that varied only by cycle (e.g. ACG versus CGA with the same number of repeats right next to each other), entirely surrounded by another SSR, or not atomic (e.g. ATAT repeated 2 times instead of AT repeated 8 times) were removed. SSRS that shared the same base and overlapped were combined into one SSR (e.g. AT repeated 8 times at position 1 and AT repeated 6 times at position 11 would be combined to AT repeated 14 times at position 1). Importantly, these filters still allow for overlapping SSRS where at least one portion of the repeating sequence overlaps another SSR.

The **Number Correct & Fixed** column is the number of SSRS that were correct or fixed. The **Percent Correct & Fixed** column is the percent of SSRS in Range that were correct or fixed. **Unique to Software** is the number of SSRS that were unique to that software. **Unique to Kmer-SSR** is the number of SSRS that were unique to Kmer-SSR.

### 4 Discussion

SSR identification is important in many biological comparisons. It is important to have 100% accuracy in SSR identification because primers often depend on the exact SSR sequence with conserved flanking sequences ([Robinson et al., 2004](#)), and phenotypic variations associated with SSRS require an accurate portrayal of a genome. Furthermore, determining the exact SSR copy number is important in species identification and aids in the identification of discrete families and individuals. Kmer-SSR fills a usability gap in SSR identification. While many SSR identification algorithms exist, it is often difficult to install, use and read the output from the algorithms available. Two of the main strengths of Kmer-SSR are its usability and the SSR filters that are easily accessible to help answer biological questions. Installing Kmer-SSR is at least as easy to install as other algorithms. Kmer-SSR was implemented in C++. It does not require any editing of the source code to find SSRS of different lengths or filter overlapping SSRS, and provides a robust documentation for its command line options. Step-by-step instructions for installation and implementation of Kmer-SSR are available with the algorithm’s source code at [http://github.com/ridgelab/Kmer-SSR](http://github.com/ridgelab/Kmer-SSR).

The filters available in Kmer-SSR help answer primary biological questions. Instead of inundating a researcher with duplicate SSRS, Kmer-SSR eliminates overlapping SSRS by only reporting the left-most SSR in each sequence when multiple SSRS are equally valid. Furthermore, longer SSRS are typically more biologically interesting, so completely enclosed SSRS are not included in the output. Importantly, these filters still allow for overlapping SSRS where at least one period size is completely outside of the previously reported SSR. These filters set Kmer-SSR apart from all other SSR identification algorithms because of its ease of use as well as its utility.

As we compared other algorithms, a few difficulties arose that made it challenging to directly compare the output from each program. We learned that QDD does not allow the sequence header line.

### Table 1. (Continued)

| Software          | CPU Time (mm:ss) | Real Time (mm:ss) | SSRS Reported | SSRS After Adjustments | SSRS In Range | Number Correct | Number Correct & Fixed | Percent Correct & Fixed | SSRS Unique to Software | SSRS Unique to Kmer-SSR |
|-------------------|-----------------|------------------|---------------|------------------------|--------------|----------------|------------------------|------------------------|--------------------------|--------------------------|
| TRF               | 1:53            | 1:53             | 223,938       | 215,818                | 22,730       | 6132           | 8192                    | 36.04                  | 0                        | 891                      | 7896                     |
| GMATo             | 0:23            | 0:23             | 23,927        | 26,743                 | 1311         | 1101           | 1101                    | 100                    | 0                        | 588                      | 887                      |
| Kmer-SSR          | 0:23            | 0:23             | 1475          | 1475                   | 1473         | 1475           | 1475                    | 100                    | NA                       | NA                       | NA                       |
| MREPS             | 0:02            | 0:02             | 2293          | 2293                   | 2293         | 1475           | 1475                    | 1475                   | 100                      | 0                        | 1475                     |
| ProGeRF           | 3:43            | 3:43             | 1065,515      | 1065,510               | 492          | 492            | 492                     | 100                    | 0                        | 988                      | 487                      |
| QDD               | 0:47            | 0:47             | 8672          | 8672                   | 1368         | 1368           | 1368                    | 100                    | 0                        | 139                      | 1336                     |
| SA-SSR            | 33:50           | 60:55            | 1430          | 1430                   | 1430         | 1430           | 1430                    | 100                    | 0                        | 57                       | 1418                     |
| SSR-Pipeline      | 3:32            | 3:32             | 124,828       | 28,20,560              | 1427         | 1427           | 1427                    | 100                    | 0                        | 73                       | 1402                     |
| SSRIT             | 0:14            | 0:14             | 12,276        | 10,386                 | 1475         | 1475           | 1475                    | 100                    | 0                        | 0                       | 1475                     |
| TRF               | 0:26            | 0:26             | 62,616        | 61,038                 | 4634         | 755            | 1242                    | 26.80                  | 0                        | 290                      | 1185                     |
| Combined          |                 |                  |               |                        |              |                |                        |                        |                          |                          |                          |
| GMATo             | 9:44            | 9:44             | 76,711,216    | 181,301                | 181,301      | 180,734        | 100                    | 72,304                 | 127,524                  |                          |                          |
| Kmer-SSR          | 9:33            | 1:18             | 199,828       | 199,828                | 199,828      | 199,828        | 199,828                 | 100                    | NA                       | NA                       | NA                       |
| MREPS             | 0:39            | 0:39             | 284,389       | 199,828                | 199,828      | 199,828        | 199,828                 | 100                    | 0                        | 199,828                  |                          |
| ProGeRF           | 87:13           | 87:13            | 44,944,317    | 140,872                | 140,753      | 140,785        | 100                    | 59,518                 | 140,310                  |                          |                          |
| QDD               | 44:45           | 44:45            | 535,085       | 192,988                | 192,988      | 192,988        | 192,988                 | 100                    | 6,997                    | 189,131                  |                          |
| SA-SSR            | 5443:20         | 1238:38          | 197,710       | 197,710                | 197,710      | 197,710        | 197,710                 | 100                    | 4190                     | 195,638                  |                          |
| SSR-Pipeline      | 4957:12         | 4957:12          | 75,275,495    | 56,381,553             | 197,587      | 197,587        | 197,587                 | 100                    | 5787                     | 194,050                  |                          |
| SSRIT             | 5:43            | 5:43             | 784,469       | 633,267                | 199,828      | 199,828        | 199,828                 | 100                    | 0                        | 199,828                  |                          |
| TRF               | 35:53           | 35:53            | 2,667,832     | 2,564,881              | 470,739      | 98,237         | 165,167                 | 35,09                  | 42,393                   | 157,435                  |                          |
to contain the vertical bar |] (and possibly other characters that have special meaning in a regular expression). Also, analysis of 1-mers in longer sequences, such as the lizard genome, exceeded 21 days in SSR-pipeline. MREPS also required pre-splitting of the input sequence files because the algorithm does not accept any characters besides A, T, C and G in the sequence lines (it will accept a very limited number of well-distributed Ns). SSRIT requires directly editing the source code to query period sizes other than lengths two through four. Similarly, QDD requires directly editing its source code to retrieve different period lengths and different SSR lengths. QDD defaults to 1-mers that must be 1 million bases long and 2-mers through 6-mers that must repeat at least 5 times. Furthermore, unlike some other algorithms, the output format for Kmer-SSR is easily parsable, and can be exported directly to an Excel spreadsheet or another tab delimited parser. GMATO, ProGeRF, SSRIT and SA-SSR have similar output formats (although, ProGeRF and SSRIT do not provide column headers). MREPS and TRF are text-based reports with embedded tables. QDD provides a semicolon-separated value report with a few fixed columns for Kmer-SSR TSV C++ K-mer Decomposition Exact 1+ 2+ X X
SA-SSR TSV C++ Combinatorial Exact 1+ 2+ X X
GMATo X TSV Perl & Java Regular Expressions Exact 1–10 2+?
MREPS Text C Combinatorial Inexact 1+ 2+
ProGeRF Web TSV Perl ? Inexact 1–12 2+
QDD SCVS Perl ? Exact 1–6 5+
SSR-Pipeline FASTA Python ? Exact 1–25 2+
SSRIT TSV Perl Regular Expressions Exact 2–4 2+
TRF X Text ? Heuristic Inexact 1+ 2+

**Table 2.** We documented each SSR algorithm’s basic usages and options based on the documentation from each algorithm.

**Note:** All algorithms can run in a Linux environment, accept command line options and take a fasta file as input. Columns in the table are as follows: GUI=Graphical user input available. The algorithms create either a text file, tab separated values (TSV), semicolon separated values (SCSV), or fasta file. The language in which the program is written is followed by the method that the algorithm uses and the type of SSRs it can find (exact or inexact). Minimum SSR period sizes and SSR repeat numbers are also listed. Finally, we list if the algorithm is multithreaded or configurable to search for specific SSRs. Only Kmer-SSR and SA-SSR are multithreaded and configurable to search for specific SSRs. The language in which the program is written is followed by the method that the algorithm uses and the type of SSRs it can find (exact or inexact). Minimum SSR period sizes and SSR repeat numbers are also listed. Finally, we list if the algorithm is multithreaded or configurable to search for specific SSRs. Only Kmer-SSR and SA-SSR are multithreaded and configurable to search for specific SSRs.

**References**

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

Chikhi, R. and Medvedev, P. (2014) Infomed and automated k-mer size selection for genome assembly. *Bioinformatics*, 30, 31–37.

Clancy, W.J. (1985) Heuristic classification. *Artif. Intell.*, 27, 289–350.

Ghandi, M. et al. (2014) Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput. Biol.*, 10, e1003711.

Han, W.-S. et al. (2007) Ranked subsequence matching in time-series databases. In: *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, p. 423-434.

Kashi, Y. et al. (1997) Simple sequence repeats as a source of quantitative genetic variation. *Trends Genet.*, 13, 74–78.

Kashi, Y. and King, D.G. (2006) Simple sequence repeats as advantageous mutators in evolution. *Trends Genet.*, 22, 253–259.

Koltsov, V. et al. (2001) Differential distribution of simple sequence repeats in eukaryotic genome sequences. *Mol. Biol. Evol.*, 18, 1161–1167.

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

Levinson, G. and Gutman, G.A. (1987) Slipped-strand mispairing: a major mechanism for DNA sequence evolution. *Mol. Biol. Evol.*, 4, 203–221.
Lopes, R.d.S. et al. (2015) ProGeRF: Proteome and Genome Repeat Finder Utilizing a Fast Parallel Hash Function. *BioMed. Res. Int.*, 2015, 1.

Meglecz, E. et al. (2014) QDD version 3.1: a user-friendly computer program for microsatellite selection and primer design revisited: experimental validation of variables determining genotyping success rate. *Mol. Ecol. Resources*, 14, 1302–1313.

Merchant, S.S. et al. (2007) The Chlamydomonas genome reveals the evolution of key animal and plant functions. *Science*, 318, 245–250.

Miller, M.P. et al. (2013) SSR_pipeline: A bioinformatic infrastructure for identifying microsatellites from paired-end Illumina high-throughput DNA sequencing data. *J. Hered.*, est056.

Murray, J. et al. (1999) Comparative sequence analysis of human minisatellites showing meiotic repeat instability. *Genome Res.*, 9, 130–136.

Pickett, B.D. et al. (2016) SA-SSR: a suffix array-based algorithm for exhaustive and efficient SSR discovery in large genetic sequences. *Bioinformatics*, 32, 2707–2709.

Robinson, A.J. et al. (2004) Simple sequence repeat marker loci discovery using SSR primer. *Bioinformatics*, 20, 1475–1476.

Temnykh, S. et al. (2001) Computational and experimental analysis of microsatellites in rice (*Oryza sativa* L.): frequency, length variation, transposon associations, and genetic marker potential. *Genome Res.*, 11, 1441–1452.

Wang, X. et al. (2013) GMATo: A novel tool for the identification and analysis of microsatellites in large genomes. *Bioinformation*, 9, 541–544.