Journal of Heredity 2013:104(6):881–885
doi:10.1093/jhered/est056
Advance Access publication September 19, 2013

Computer Note

SSR_pipeline: A Bioinformatic Infrastructure for Identifying Microsatellites From Paired-End Illumina High-Throughput DNA Sequencing Data

MARK P. MILLER, BRIAN J. KNAUS, THOMAS D. MULLINS, AND SUSAN M. HAIG

From the U.S. Geological Survey, Forest and Rangeland Ecosystem Science Center, 3200 SW Jefferson Way, Corvallis, OR 97331 (Miller, Mullins, and Haig); and the Horticultural Crop Research Unit, USDA-ARS, Corvallis, OR (Knaus).

Address correspondence to Mark P. Miller at the address above, or e-mail: mpmiller@usgs.gov.

Data deposited at Dryad: http://dx.doi.org/10.5061/dryad.n65k2

SSR_pipeline is a flexible set of programs designed to efficiently identify simple sequence repeats (e.g., microsatellites) from paired-end high-throughput Illumina DNA sequencing data. The program suite contains 3 analysis modules along with a fourth control module that can automate analyses of large volumes of data. The modules are used to 1) identify the subset of paired-end sequences that pass Illumina quality standards, 2) align paired-end reads into a single composite DNA sequence, and 3) identify sequences that possess microsatellites (both simple and compound) conforming to user-specified parameters. The microsatellite search algorithm is extremely efficient, and we have used it to identify repeats with motifs from 2 to 25 bp in length. Each of the 3 analysis modules can also be used independently to provide greater flexibility or to work with FASTQ or FASTA files generated from other sequencing platforms (Roche 454, Ion Torrent, etc.). We demonstrate use of the program with data from the brine fly *Ephydra packardi* (Diptera: Ephydridae) and provide empirical timing benchmarks to illustrate program performance on a common desktop computer environment. We further show that the Illumina platform is capable of identifying large numbers of microsatellites, even when using unenriched sample libraries and a very small percentage of the sequencing capacity from a single DNA sequencing run. All modules from *SSR_pipeline* are implemented in the Python programming language and can therefore be used from nearly any computer operating system (Linux, Macintosh, and Windows).

Key words: *next-generation DNA sequencing, Python, simple sequence repeat*

Next-generation sequencing platforms have opened the door to numerous technological and theoretical advances for the study of natural populations. Reducing time and financial costs while simultaneously increasing the volume of data generated by multiple orders of magnitude, sequence data generated by these platforms have broad utility and can be used to address numerous research topics (Ekblom and Galindo 2011; Haig et al. 2011; Harrison and Kidner 2011; Funk et al. 2012). Despite the appeal of using high-throughput DNA sequencing platforms to address research questions in ecology and evolutionary biology, we suggest that more conventional approaches, such as use of microsatellite loci, are nonetheless likely to remain common into the future. Microsatellite loci are broadly applicable to the fields of molecular ecology, evolutionary biology, and conservation (Schlötterer and Pemberton 1998) and are often more than adequate for addressing questions related to genetic structure or population history (Selkoe and Toonen 2006). They can also excel in situations where interindividual relationships need to be identified (Jones and Ardren 2003) or for forensic investigations (Alacs et al. 2010). Furthermore, we note that capillary electrophoresis-based instruments, such as ABI 3100 or 3730 style machines, remain commonly available worldwide and allow researchers to perform microsatellite genotyping of individuals using well-established and relatively inexpensive protocols. The availability of these instruments and protocols will likely ensure that microsatellite loci will continue to be commonplace for the foreseeable future.

Identification and development of microsatellites was previously considered to be a time-consuming task (Glenn and Schable 2005). However, because of the continued demand for microsatellite loci, next-generation sequencing platforms have become useful for circumventing many of the laborious stages of the microsatellite marker development process (Allentoft et al. 2009; Mikheyev et al. 2010; Jennings et al. 2011; Castoe et al. 2012; Zalapa et al. 2012). Given the large volumes of DNA sequence data that are generated from high-throughput sequencing runs, computer programs can be used to efficiently identify and characterize microsatellite DNA using various computational approaches (see Du et al. 2013 for a comparison of different programs). In this article, we describe a new computational infrastructure that is designed to identify microsatellite sequences. Our software differs from other programs in that it is optimized for paired-end Illumina sequence reads in FASTQ format, the de facto standard for high-throughput sequence data (Cock et al. 2010). However, the program's modular design permits flexibility to deal with FASTQ formatted sequences produced by other high-throughput sequencing platforms (e.g., Roche 454, Ion Torrent, etc.). A summary of the program's design, implementation, and performance is provided below.

Program Description

SSR_pipeline is a small suite of command line programs based on 3 separate analysis modules plus a fourth controller module that can be used for analysis automation. The programs accept FASTQ (and in some cases FASTA) files as input. Given the large volume of data produced by the Illumina platform (and all next-generation sequencing platforms in general), the program can optionally accept input files as compressed archives (gzip files) in order to reduce computer storage requirements. The program is ideally designed for analyses of paired-end sequences from low-coverage whole genomic DNA libraries where the sequenced fragments have been sized to facilitate a paired-end alignment step (see Benchmarking and Code Performance for an example). Data from reduced sequence representation approaches (e.g., RAD sequencing; Miller et al. 2007) are also appropriate if paired-end sequences are generated. Individual program modules can also be applied outside of the *SSR_pipeline* framework, as described in more detail below.

Module *SSR_pipeline* is designed to automate the operation of the other 3 modules contained in the program. It is provided to allow users to go quickly and easily from raw Illumina DNA sequence data to sets of candidate microsatellite loci (Figure 1). Analyses will generally follow a 3-step process that can be controlled and parameterized using a simple configuration file (see Supplementary Material online). First, module *quality_sort* is used to process raw FASTQ files that are output from Illumina runs. After a sequencing run, the Illumina *CASSAVA* software automatically performs a quality assessment step for each sequence generated. However, all sequence data (even those with failed quality grades) are included in the output files, and good versus bad sequences are identified only by a binary flag included in each FASTQ sequence header. Thus, *quality_sort* reads each sequence header to identify high-quality sequences and preserves only the subset of paired-end reads where sequence pairs have passed quality standards. This module may also be used independently outside of the *SSR_pipeline* framework for general paired-end Illumina data filtering purposes (e.g., before performing a complete genome assembly, etc.).

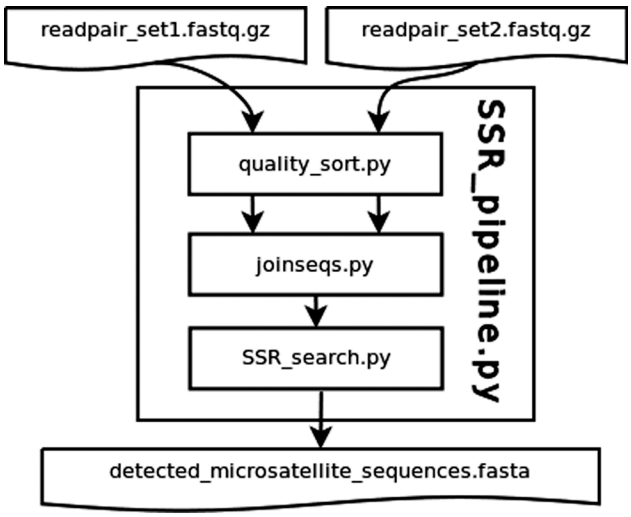

Figure 1. Overview of workflow associated with using *SSR_pipeline* to identify microsatellite DNA. The procedure involves 1) quality filtering of paired-end reads, 2) alignment of paired-end reads, and 3) searches for microsatellites that conform to user-specified parameters. The figure provides an overview of the complete analysis pipeline; however, individual modules can be used independently from one another for more customized analyses. See Program Description for more details.

Next, module *joinseqs* analyzes the output from *quality_sort* and aligns the 2 sequences from each paired-end read to produce a longer single DNA sequence. An additional extension to this module (*joinseqs_ext*) contains a set of compiled functions that provide substantial performance enhancements. The algorithm used by *joinseqs* is essentially a Python-based implementation of the FLASH sequence alignment protocol, which possesses excellent performance and accuracy attributes (Magoč and Salzberg 2011). User-defined parameters for the alignment procedure include the minimum acceptable overlap length, the maximum proportion of mismatches allowed within the best overlap, and the maximum sequence overlap beyond which a penalized mismatch statistic is calculated (Magoč and Salzberg 2011). Outside of *SSR_pipeline*, *joinseqs* can be independently applied in any analysis workflow desired and may be a particularly effective replacement for the original FLASH program, especially if developers wish to use the Python environment for their work.

Finally, module *SSR_search* is used to analyze the output from *joinseqs* and identify the set of DNA sequences that contain microsatellites. It is based on an extremely efficient and flexible search algorithm that theoretically allows for detection of any microsatellite motif size desired. In practice, we experienced memory limitations on 32-bit systems when trying to detect motifs larger than 25 bp in size. These limitations are reduced on 64-bit systems. Users may specify search constraint parameters that determine repeat lengths, the minimum number of desired repeats, and lengths of terminal flanking regions. The latter parameter can be used to help ensure that microsatellites are located between flanking regions of sufficient length and provide space for primer design. All sequences identified with *SSR_search* are annotated with information that summarizes sequence motifs, repeat length, and sequence positions of repeats. The program also detects compound repeats when present. In addition to being used in the context of *SSR_pipeline*, the *SSR_search* module has broader utility in that it can independently be used to analyze sequence data from any FASTA or FASTQ file, thus extending its utility beyond just the Illumina sequencing platform for which it was originally designed.

Many programs that identify microsatellite sequences also automate the polymerase chain reaction (PCR) primer design process (see summary in Du et al. 2013). Because of the large number of microsatellites that can be identified with our approach (see Benchmarking and Code Performance), we are of the opinion that a quick "manual evaluation" step is more useful than an automatic switch to generating PCR primers.

In practice, after performing analyses of a high-throughput sequencing data set, we find it easiest to spend 5–10 min visually inspecting the output from the *SSR_search* module. Because of the way that the program annotates the sequence headers, it is a simple process to quickly identify a few hundred sequences that appear to be more conducive to PCR primer design and that possess desirable features (i.e., longer simple repeats as opposed to shorter repeats or compound repeats). Once that small subset of sequences is identified, existing online resources such as BatchPrimer3 (You et al. 2008) or PrimerQuest (http://www.idtdna.com/Primerquest/Home/Index [last accessed 26 August 2013]) can be used for PCR primer design.

Benchmarking and Code Performance

We illustrate the performance of *SSR_pipeline* using data from the brine fly *Ephydra packardi* (Diptera: Ephydridae). An indexed genomic library was prepared using a TruSeq DNA library preparation kit (Illumina). Genomic DNA was fragmented using a Bioruptor sonicator (Diagenode, Inc.), and a 200-bp slice of gel was subsequently extracted from a 1% agarose gel. Unlike other studies (e.g., Jennings et al. 2011), no enrichment for microsatellite DNA was performed. The *E. packardi* library was 1 of 21 separate equimolar libraries included within a single lane of an Illumina HiSeq2000 flow cell. Thus, the sequencing effort for *E. packardi* represented ~4.7% of the sequencing capacity of a single lane or ~0.59% of the capacity of the full flow cell. The *E. packardi* sequence data were generated using a paired-end HiSeq 2000 Illumina run (100-bp reads) and resulted in 9 015 656 raw paired-end reads once the sequencing run was completed (Short-Read Archive # SRA099359).

Table 1 presents empirical benchmarks that illustrate typical performance and outcomes of analyses with *SSR_pipeline*. The analyses were performed on a Dell Optiplex 755 desktop workstation (3 GHz Intel Core 2 Duo CPU with 2 GB of RAM) running 32-bit Microsoft Windows XP with Service Pack 3. Parameters for the paired-end alignment step included a minimum 10-bp overlap, maximum mismatch ratio of 0.25, and a maximum overlap that incurred a penalized mismatch ratio of 70 bp. These general parameter values work well for alignment of 100-bp paired-end reads (Magoč and Salzberg 2011). Of the more than 9 million paired-end reads introduced into the analysis, 91% were preserved after evaluation with module *quality_sort*. Approximately 4.9 million reads (60% of sequences that passed the quality assessment step) were successfully aligned using module *joinseqs*. This percentage is consistent with the ~200-bp fragments selected during library construction.

Parameters for the microsatellite searches are listed in Table 1 and included the additional imposition of 40-bp flanking regions as a search constraint. Results of these searches demonstrate that large numbers of candidate microsatellite loci can be readily identified using *SSR_pipeline* and that the analyses are easily capable of running in a matter of minutes on commonly available computer environments. Even when using a small fraction of the full sequencing capacity of the Illumina platform, we still identified >60 000 sequences that contained microsatellites, with the majority of those sequences containing di-, tri-, and tetranucleotide motifs that are most commonly used for genotyping purposes. Of these sequences, more than 10 000 contained microsatellites at locations between 40-bp flanking regions. This subset of sequences may be ideally targeted for formal microsatellite

Table 1 Benchmark data and numerical examples of outcomes from *SSR_pipeline* analyses of an Illumina DNA sequence data set comprised of 9 015 656 paired-end reads from *Ephydra packardi* (Diptera: Ephydridae)

		Analysis time (min:s)	
	Number of reads	Compressed (gzipped) input files	Uncompressed input files
Initial number of read-pairs	9 015 656	n.a.	n.a.
Read-pairs passing quality standards	8 209 173	32:01	09:08
Joined read-pairs (FLASH algorithm)	4 919 005	33:42	14:11
Microsatellite search			
2-mers: min. repeats = 7	16 278 (6944)	02:45	02:08
3-mers: min. repeats = 6	4824 (1821)	02:47	02:06
4-mers: min. repeats = 5	24 865 (4608)	02:52	02:16
5-mers: min. repeats = 4	9231 (714)	02:52	02:18
6-mers: min. repeats = 4	2766 (152)	02:57	02:20
7-mers: min. repeats = 4	153 (26)	02:57	02:19
8-mers: min. repeats = 4	3260 (608)	03:02	02:27
9-mers: min. repeats = 4	168 (38)	03:06	02:24
10-mers: min. repeats = 4	230 (28)	03:10	02:30
25-mers: min. repeats = 2	216 (85)	05:17	03:07

Analyses were performed using an Intel Core 2 Duo (3 GHz) desktop computer running 32-bit Microsoft Windows XP. In the "Microsatellite search" section, values listed under the "number of reads" column reflect the total number of sequences out of 4 919 005 that contained the specified microsatellite type. Adjacent values in parentheses list the number of sequences where microsatellites were located between 40-bp flanking sequences to facilitate primer design for downstream PCR-based analyses. We further present benchmark timings when input and output files from analyses were stored in compressed versus native forms. n.a., not applicable.

locus development, as the presence of the 40-bp flanking regions may help facilitate design of PCR primers for genotyping assays.

Data in Table 1 also highlight the tradeoff between program execution speed and use of compressed (gzipped) versus uncompressed files. Program execution is slower when analyzing compressed files because of the computational overhead required to dynamically extract and uncompress data. However, use of compressed files may be advantageous if data storage space is limited. For example, the uncompressed FASTQ data associated with the 9 015 656 paired-end reads used in the first stage of the analysis requires 4.4 GB of storage space, and downstream uncompressed files generated by *SSR_pipeline* would require space for an additional 7.5 GB of information. By contrast, the input files would require 1.5 GB of storage if compressed, and *SSR_pipeline*'s output would only generate an additional 2.6 GB of compressed data after the analysis is completed.

Comparison with Other Programs

SSR_pipeline is unique in that it is optimized for use with a particular sequencing platform (paired-end Illumina runs). To our knowledge, no other program combines its capabilities (sequence quality sorts, paired-end alignments, and microsatellite searches) into a single automated pipeline. However, numerous programs exist that can facilitate identification of microsatellite sequences. Given that module *SSR_search* can be used independently of the full automated pipeline, we provide a summary of the modules capabilities in comparison to other programs that can perform similar analyses. The following summary of *SSR_search* extends Table 1 of Du et al. (2013), which provides an excellent overview of 9 different microsatellite detection programs: *SSR_search* is capable of finding simple and compound microsatellites but will not identify imperfect microsatellite sequences. It is written in the Python programming language and can be used under most common operating system (Windows, Linux, and Mac). The program is run from a command line/console user interface and includes features that allow it to batch-process arbitrary numbers of file sets. Lengths of terminal flanking sequences can be specified as an analysis parameter to facilitate downstream primer development, and basic summary statistics of the microsatellite search procedure are recorded in text-based log files. The program does not create formal databases that contain search results nor does the program automate the process of interfacing with Primer3 (Rozen and Skaletsky 2000).

Implementation and Availability

SSR_pipeline is implemented in the Python programming language (www.python.org), making it capable of running on most common computing platforms (Linux, Macintosh, and Windows). The program, its documentation, and sample data files can be downloaded from http://pubs.usgs.gov/ds/778/ [last accessed 26 August 2013]. The Python code from the *joinseqs* module is enhanced for performance speed by a compiled extension module created using Cython (Behnel et al. 2011; www.cython.org [last accessed 26 August 2013]). Instructions for compiling this extension module with freely available tools (the Gnu Compiler Collection or Microsoft Visual C++ Express) are provided in the documentation. The pure Python modules combined with a copy of the self-compiled extension module will be the preferred choice for most users. However, for users who are unable to compile the extension module or are unable to install Python, we have also made available sets of prepackaged executable files created using PyInstaller (www.pyinstaller.org [last accessed 26 August 2013]). These executables contain all Python scripts, extension modules, and a minimal installation of any necessary Python runtime libraries in a single file. The Windows executables were created using Python 2.7.3 under 32-bit Windows XP and can be used under any later Windows version (we have tested with 32-bit Windows XP and Vista and 64-bit Windows 7 without any issues). 32-bit Linux executables are available that were built under CentOS 5.8 using Python 2.4 and glibc (Gnu C Library) version 2.5. These should work with any 32-bit Linux as long as it is using glibc version 2.5 or later. 64-bit Linux executables were built under CentOS 6.2 using Python 2.6.6 and glibc version 2.12. 64-bit OS X binaries were built under OSX 10.8.2 using Apple's system Python (version 2.7.2) and are compatible with OS X 10.6 and later.

Supplementary Material

Supplementary material can be found at http://www.jhered.oxfordjournals.org/.

Funding

U.S. Geological Survey, Forest and Rangeland Ecosystem Science Center; the U.S. Forest Service.

Acknowledgments

We thank R. Cronn and T. Jennings for helpful discussions in early phases of software development. R. Cornman provided helpful comments on an earlier manuscript draft. Any use of trade, product, or firm names is for descriptive purposes only and does not imply endorsement by the US Government.

References

Alacs EA, Georges A, FitzSimmons NN, Robertson J. 2010. DNA detective: a review of molecular approaches to wildlife forensics. Forensic Sci Med Pathol. 6:180–194.

Allentoft M, Schuster SC, Holdaway RN, Hale ML, McLay E, Oskam C, Gilbert MTP, Spencer P, Willerslev E, Bunce M. 2009. Identification of microsatellites from an extinct moa species using high-throughput (454) sequence data. Biotechniques. 46:195–200.

Behnel S, Bradshaw R, Citro C, Dalcin L, Seljebotn DS, Smith K. 2011. Cython: the best of both worlds. Comput Sci Eng. 13:31–39.

Castoe TA, Poole AW, de Koning AP, Jones KL, Tomback DF, Oyler-McCance SJ, Fike JA, Lance SL, Streicher JW, Smith EN, et al. 2012. Rapid microsatellite identification from Illumina paired-end genomic sequencing in two birds and a snake. PLoS One. 7:e30953.

Cock PJA, Fields CJ, Goto N, Heuer ML, Rice PM. 2010. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Res. 38:1767–1771.

Du L, Li Y, Zhang X, Yue B. 2013. MSDB: a user-friendly program for reporting distribution and building databases of microsatellites from genome sequences. J Hered. 104:154–157.

Ekblom R, Galindo J. 2011. Applications of next generation sequencing in molecular ecology of non-model organisms. Heredity. 107:1–15.

Funk WC, McKay JK, Hohenlohe PA, Allendorf FW. 2012. Harnessing genomics for delineating conservation units. Trends Ecol Evol. 27:489–496.

Glenn TC, Schable NA. 2005. Isolating microsatellite DNA loci. Methods Enzymol. 395:202–222.

Haig SM, Bronaugh W, Crowhurst R, D'Elia J, Eagles-Smith C, Epps C, Knaus B, Miller MP, Moses M, Oyler-McCance S, et al. 2011. Applications of genetics in avian conservation. Auk. 128:205–229.

Harrison N, Kidner CA. 2011. Next-generation sequencing and systematics: what can a billion base pairs of DNA sequence data do for you? Taxon. 60:1552–1566.

Jennings TN, Knaus BJ, Mullins TD, Haig SM, Cronn RC. 2011. Multiplexed microsatellite recovery using massively parallel sequencing. Mol Ecol Resour. 11:1060–1067.

Jones AG, Ardren WR. 2003. Methods of parentage analysis in natural populations. Mol Ecol. 12:2511–2523.

Magoč T, Salzberg SL. 2011. FLASH: fast length adjustment of short reads to improve genome assemblies. Bioinformatics. 27:2957–2963.

Mikheyev AS, Vo T, Wee B, Singer MC, Parmesan C. 2010. Rapid microsatellite isolation from a butterfly by de novo transcriptome sequencing: performance and a comparison with AFLP-derived distances. PLoS One. 5:e11212.

Miller MR, Dunham JP, Amores A, Cresko WA, Johnson EA. 2007. Rapid and cost-effective polymorphism identification and genotyping using restriction site associated DNA (RAD) markers. Genome Res. 17:240–248.

Rozen S, Skaletsky HJ. 2000. Primer3 on the WWW for general users and for biologist programmers. In: Krawetz S, Misener S, editors. Bioinformatics methods and protocols: methods in molecular biology. Totowa (NJ): Humana Press. p. 365–386.

Schlötterer C, Pemberton J. 1998. The use of microsatellites for genetic analysis of natural populations—a critical review. In: DeSalle R, Schierwater B, editors. Molecular approaches to ecology and evolution. Basel (Switzerland): Birkhäuser Verlag. p. 71–86.

Selkoe KA, Toonen RJ. 2006. Microsatellites for ecologists: a practical guide to using and evaluating microsatellite markers. Ecol Lett. 9:615–629.

You FM, Huo N, Gu YQ, Luo M-C, Ma Y, Hane D, Lazo GR, Dvorak J, Anderson OD. 2008. BatchPrimer3: a high throughput web application for PCR and sequencing primer design. BMC Bioinformatics. 9:253.

Zalapa JE, Cuevas H, Zhu H, Steffan S, Senalik D, Zeldin E, McCown B, Harbut R, Simon P. 2012. Using next-generation sequencing approaches to isolate simple sequence repeat (SSR) loci in the plant sciences. Am J Bot. 99:193–208.

Received April 18, 2013; First decision May 16, 2013; Accepted August 2, 2013

Corresponding Editor: Howard Ross