Bioinformatics, 33(5), 2017, 751–753
doi: 10.1093/bioinformatics/btw741
Advance Access Publication Date: 16 December 2016
Applications Note

OXFORD

Sequence analysis

SeqLib: a C++ API for rapid BAM manipulation, sequence alignment and sequence assembly

Jeremiah Wala[1,2,3] and Rameen Beroukhim[1,2,3,*]

[1]The Broad Institute of Harvard and MIT, Cambridge, MA 02142, USA, [2]Bioinformatics and Integrative Genomics, Harvard University, Cambridge, MA 02138, USA and [3]Department of Cancer Biology, Dana-Farber Cancer Institute, Boston, MA 02115, USA

*To whom correspondence should be addressed.
Associate editor: Inanc Birol

Received and revised on October 7, 2016; editorial decision on November 16, 2016; accepted on November 18, 2016

Abstract

We present SeqLib, a C++ API and command line tool that provides a rapid and user-friendly interface to BAM/SAM/CRAM files, global sequence alignment operations and sequence assembly. Four C libraries perform core operations in SeqLib: HTSlib for BAM access, BWA-MEM and BLAT for sequence alignment and Fermi for error correction and sequence assembly. Benchmarking indicates that SeqLib has lower CPU and memory requirements than leading C++ sequence analysis APIs. We demonstrate an example of how minimal SeqLib code can extract, error-correct and assemble reads from a CRAM file and then align with BWA-MEM. SeqLib also provides additional capabilities, including chromosome-aware interval queries and read plotting. Command line tools are available for performing integrated error correction, micro-assemblies and alignment.

Availability and Implementation: SeqLib is available on Linux and OSX for the C++98 standard and later at github.com/walaj/SeqLib. SeqLib is released under the Apache2 license. Additional capabilities for BLAT alignment are available under the BLAT license.

Contact: jwala@broadinstitue.org; rameen@broadinstitute.org

1 Introduction

Extracting signals from sequencing data typically requires some combination of three core operations: sequence access (reading and writing), sequence alignment (or re-alignment) and sequence assembly. The accuracy of these operations is critical to many sequence analysis methods, and typically comprises the bulk of the computational requirements. In practice, these operations are often split among several tools, with connections between them requiring separate connectors and heavy reading and writing from disk. An improved solution would integrate these core operations into a single framework, allowing rapid manipulations of sequencing data in any form without additional CPU, disk or memory overhead. Such an approach would also facilitate the development of more advanced tools that could integrate alignment or assembly without requiring extensive knowledge of the implementations of each.

Although a number of APIs are available for manipulating sequencing data, there is still a need for an integrated library capable of all three core sequence analysis operations. Notable packages for accessing sequencing data and/or performing local sequence alignment include Pysam (github.com/pysam-developers/pysam) for Python, Rsamtools for R (Morgan, 2016) and Htsjdk (github.com/samtools/htsjdk) for Java. For the C++ developer, BamTools (Barnett *et al.*, 2011) and SeqAn (Döring *et al.*, 2008) have been indispensible for hundreds of bioinformatics projects.

We present SeqLib, a C++ API and command line tool that integrates sequence access, global/local sequence alignment, and sequence assembly into a single library. We highlight two distinct advantages of SeqLib over existing C++ sequence analysis APIs: (1) improved computational and memory performance for BAM/CRAM/SAM reading and writing through our integration with HTSlib (github.com/samtools/htslib), and (2) in-memory access to BWA-MEM (Li, 2015a) and BLAT (Kent, 2002) for sequence alignment and to Fermi (Li, 2012) for error correction and sequence assembly. By integrating with existing and widely supported C

Table 1. SeqLib capabilities and implementing C++ class

Capability	Class
Multi BAM/CRAM/SAM read	BamReader
BAM/CRAM/SAM write	BamWriter
SAM record and operations	BamRecord
Reference genome queries	RefGenome
BWA-MEM indexing/alignment	BWAWrapper
BLAT alignment	BLATWrapper
Bloom filter error correction (BFC)	BFC
Assembly and error correction	FermiAssembler
Multi-criteria read filtering	ReadFilter
ASCII alignment plotting	SeqPlot
Interval queries and operations	GenomicRegionCollection

projects, SeqLib naturally incorporates future advancements without requiring new development by the end-user. SeqLib also integrates additional capabilities, including a chromosome-aware interval tree (github.com/ekg/intervaltree), an API for read filtering (Wala *et al.*, 2016) and lightweight sequence alignment plots. SeqLib is designed to be simple to use, allowing developers to focus on novel bioinformatics and scientific solutions rather than on software optimization and memory management.

2 Features and methods

Core functionality and implementing classes are listed in Table 1.

BAM/SAM/CRAM access

SeqLib supports BAM, SAM and CRAM files for reading and writing, as well as streaming with standard input and output. Simultaneous streaming from multiple coordinate-sorted files is also supported. Alignment records are stored internally using a maximally compact format provided by HTSlib, which compresses sequences into 4-bits per base pair. The operations on SAM records provided in SeqLib directly modify this compact structure, providing a low memory footprint and native integration with other HTSlib functions. Memory allocation/deallocation is handled internally by the API.

Sequence alignment and assembly

BWA-MEM and BLAT provide the core operations for sequence alignments in SeqLib. In addition to in-memory global alignments to a reference genome, SeqLib provides for BWA indexing and queries on run-time generated sequences, such as assembled contigs. Run-time indexing operations are particularly useful for local sequence alignments to newly generated target sequences, where traditional dynamic programming approaches can be slow.

Sequence assemblies in SeqLib are available through direct access to Fermi string graph assemblies. Bloom filter-based sequence error correction using BFC (Li, 2015b) is available as part of the assembly process or as a stand-alone unit. We provide here an example of how assembly with Fermi can be combined with sequence access and sequence alignment with BWA-MEM to create a local-assembly and realignment tool with minimal code:

```
BamReader br;
br.Open("in1.cram"); //open a CRAM file
BamRecord r;
FermiAssembler f;  //create a Fermi assembler
while(br.GetNextRecord(r))//import reads
f.AddRecord(r);  //add read to assembler
f.CorrectReads();     //error correct with BFC
f.PerformAssembly(); //assemble with Fermi
BWAWrapper b;   //create a bwa aligner
b.LoadIndex("hg19.fa"); //load reference genome
BamRecordVector results;   //store contig alignments
for(auto& i: f.GetContigs())
b.AlignSequence(i.name, i.seq, results);//align
```

Table 2. Memory and computational performance for reading and storing 5 million 101 base pair reads

Method	Memory (Gb)	CPU (s)
SeqLib	3.15	11.77
SeqLib (with string data)	3.86	14.33
BamTools	6.93	17.52
BamTools (with string data)	13.32	48.74
SeqAn	8.35	92.39

SeqLib and BamTools both have an optional step to decompress the sequence data into a full C++ string as needed.

Command line tool

SeqLib provides a command-line tool for performing combinations of targeted assemblies, error correction and BWA-MEM realignments. For instance, assembly and contig alignment of variant sites from a VCF or BED file can be accomplished with a single command, or as part of a chain of piped operations.

3 Results and discussion

We compared the memory and computational performance of SeqLib to SeqAn and BamTools by reading and storing 5 million 101 base-pair reads from a BAM file (Table 2). We further tested against the BamTools 'core' option, which only loads non-string data, and SeqLib with an additional decompression of the 4 bit compressed sequence into a C++ string. SeqLib showed improved computational and memory performance, and was able to access 5 million reads in under 20 s. All three APIs required fewer than 10 lines of code each to implement this test. Each API was able to randomly access 1000 regions in a BAM file in a similar time (SeqLib: 10.4 s, BamTools: 13.3 s, SeqAn: 9.3 s).

To evaluate the performance of SeqLib in a more complex environment, we re-built FreeBayes (Garrison and Marth, 2012) to use SeqLib. On a 1 Gb BAM, the SeqLib and original versions of FreeBayes each reported identical variants. The runtime for the original FreeBayes was 1249 s versus 344 s for the SeqLib version. The SeqLib version of FreeBayes also gained support for CRAM files.

By combining some of the most widely used and trusted sequence access, alignment and assembly tools into a single API, we provide an intuitive and powerful framework for developing efficient and integrated bioinformatics tools in C++. Full documentation, unit testing and a number of example use-cases are available at: https://github.com/walaj/SeqLib.

Acknowledgements

We would like to thank Heng Li for helpful comments and as the primary developer of BWA-MEM, HTSlib, Fermi and BFC, Jim Kent as the developer of BLAT, Erik Garrison for his interval tree implementation and Steve Huang and Steve Schumacher for comments and suggestions to the code.

Funding

National Institutes of Health (T32 HG002295/HG/NHGRI, U54CA143798 and R01CA188228), DFCI-Novartis Drug Discovery Program, and the Cure Starts Now Foundation.

Conflict of interest: none declared.

References

Barnett,D. *et al*. (2011) BamTools: a C++ API and toolkit for analyzing and managing BAM files. *Bioinformatics*, **27**, 1691–1692.

Döring,A. *et al*. (2008) SeqAn An efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**, 11–19.

Garrison,E. and Marth,G. (2012) Haplotype-based variant detection from short-read sequencing. *arXiv*, q-bio.GN.

Kent,W. (2002) BLAT–the BLAST-like alignment tool. *Genome Res*., **12**, 656–664.

Li,H. (2015a) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv*, 1–3.

Li,H. (2015b) BFC: correcting Illumina sequencing errors. *Bioinformatics*, **31**, 2885–2887.

Li,H. (2012) Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. *Bioinformatics*, **28**, 1838–1844.

Wala,J. *et al*. (2016) VariantBam: filtering and profiling of next-generational sequencing data using region-specific rules. *Bioinformatics*, **32**, 2029–2031.

Morgan,M. *et al*. (2016). Rsamtools: Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import. R package version 1.24.0, bioconductor.org/packages/release/bioc/html/Rsamtools.html.