BIOINFORMATICS APPLICATIONS NOTE

Vol. 27 no. 12 2011, pages 1691–1692
doi:10.1093/bioinformatics/btr174

Genome analysis

Advance Access publication April 14, 2011

# BamTools: a C++ API and toolkit for analyzing and managing BAM files

Derek W. Barnett[1,*], Erik K. Garrison[1], Aaron R. Quinlan[2], Michael P. Strömberg[3] and Gabor T. Marth[1]

[1]Department of Biology, Boston College, Chestnut Hill, MA 02467, [2]Department of Public Health Sciences and Center for Public Health Genomics, University of Virginia School of Medicine, Charlottesville, VA 22908 and [3]Illumina, Inc., San Diego, CA 92121, USA

Associate Editor: Alfonso Valencia

**ABSTRACT**

**Motivation:** Analysis of genomic sequencing data requires efficient, easy-to-use access to alignment results and flexible data management tools (e.g. filtering, merging, sorting, etc.). However, the enormous amount of data produced by current sequencing technologies is typically stored in compressed, binary formats that are not easily handled by the text-based parsers commonly used in bioinformatics research.

**Results:** We introduce a software suite for programmers and end users that facilitates research analysis and data management using BAM files. BamTools provides both the first C++ API publicly available for BAM file support as well as a command-line toolkit.

**Availability:** BamTools was written in C++, and is supported on Linux, Mac OSX and MS Windows. Source code and documentation are freely available at http://github.org/pezmaster31/bamtools.

**Contact:** barnetde@bc.edu

Received on December 21, 2010; revised on March 23, 2011; accepted on March 31, 2011

## 1 INTRODUCTION

The 1000 Genomes Project created the Sequence Alignment/Map (SAM) format and its sister format, the Binary Alignment/Map (BAM), to provide a 'generic and modular approach to the analysis of genomic sequencing data' (Li *et al.*, 2009). Such formats are necessary to provide a standardized vehicle for reporting alignment results and analyzing them using a wide variety of tools. The binary, compressed nature of BAM has made it the format of choice in many large-scale sequencing projects—where the text-based SAM does not scale to the massive amounts of data produced. However, storing the data in binary, compressed form comes at a cost as well. Casual programmers and end-users can no longer use the text-based parsing techniques to which they are typically accustomed. APIs and tools that can accomplish similar tasks, while managing BAM's more complex data storage behind the scenes, are vital to genomics researchers. BamTools is a flexible, efficient and easy-to-use suite designed to serve just these sorts of operations.

*To whom correspondence should be addressed.

## 2 FEATURES AND METHODS

### 2.1 The API

The BamTools API provides programmers with intuitive interfaces for querying and generating BAM files. The primary classes used by client code include BamReader, BamWriter and the BamAlignment data structure. A few additional modules exist as convenience classes: BamMultiReader allows synchronized reading from several BAM files, SamHeader provides direct query and modification of the SAM-formatted header text that is included in BAM files, and BamIndex, which serves as an interface hook for advanced clients to implement their own custom index schemes. While APIs for processing BAM files exist for other languages (Li *et al.*, 2009; McKenna *et al.*, 2010), to our knowledge the BamTools library is the only C++-specific BAM API freely available at the time of writing. By offering a BAM API implemented in C++, we provide the large community of C++ developers a tool that can leverage that language's main advantage—combining raw performance with the benefits of object-oriented design. These include clearly labeled data structures, intuitive class interfaces and utilizing the RAII (Resource Acquisition Is Initialization) idiom, which makes client code simpler and safer. The following example illustrates how to create a new BAM file containing only high-quality alignments from the forward strand of a reference sequence:

```
BamReader reader;
BamWriter writer;
// .. open reader & writer with desired files

BamAlignment al;
while ( reader.GetNextAlignment(al) ) {
    if ( !al.IsReverseStrand() && al.MapQuality >= 75 )
        writer.SaveAlignment(al);
}
```

We tested the raw read-through time on a BAM file containing the reads from 56 CEU samples from the 1000 Genomes low-coverage Pilot Project (The 1000 Genomes Project Consortium, 2010), resulting in 116-fold coverage of chromosome 20. It took on average 5:32 min to read every alignment sequentially from this file, on a single CPU (corresponding to ~4.5 h for a whole-genome file, at the same coverage). Resource usage is ultimately application dependent; however, the basic process of reading alignments from BAM files is I/O bound rather than CPU bound.

### 2.2 The toolkit

The BamTools command-line toolkit provides end-users with a suite of utilities for querying and manipulating BAM files. Table 1 offers a subset of the currently available tools. There is some overlap of features found in the BamTools and SAMtools suites. However, it is not our intention to attempt to replace SAMtools. Instead, BamTools provide features that extend the flexibility of next-generation sequencing analysis. These features include an

**Table 1.** BamTools command-line toolkit

| Utility | Description |
|---|---|
| convert | Converts between BAM and a number of other formats. |
| count | Prints number of alignments in BAM file(s). |
| coverage | Prints coverage information from a BAM file. |
| filter | Filters BAM file(s) based on user-specified criteria. |
| header | Prints BAM header information. |
| index | Generates index for BAM file (either BAI or BTI). |
| merge | Merges multiple BAM files into single file. |
| sort | Sorts the BAM file. |
| split | Splits a BAM file into multiple files, based on some criteria. |
| stats | Prints general statistics from input BAM file(s). |

alternative indexing format, basic coverage output, conversion of alignment data to other text formats (e.g. BED, JSON, YAML), the ability to split a BAM into multiple files based on some criteria (e.g. reference, read group, mapped status) in a single command and a more comprehensive filtering scheme.

*2.2.1 Scriptable filtering* The filter utility in BamTools provides a powerful scripting feature that allows a user to create complex filter operations. While various toolkits offer some level of filtering capability, the ability to utilize combinations of both AND and OR logic is a novel and useful feature. The script is based on JavaScript Object Notation (JSON), providing intuitive named fields to define filter properties and an optional 'rule'. The example script shown below will result in an output BAM file containing only alignments that have high map quality OR both mates mapped (inclusive), while excluding alignments from read groups starting with the pattern 'ERR':

```
{ "filters" : [
      { "id" : "inAnyErrorReadGroup",
        "tag" : "RG:ERR*"
      },
      { "id" : "highMapQuality",
        "mapQuality" : ">=75"
      },
      { "id:" : "bothMatesMapped",
        "isMapped" : "true",
        "isMateMapped" : "true"
      }
    ],
  "rule" : "!inAnyErrorReadGroup &
            (highMapQuality | bothMatesMapped)"
}
```

## 2.3 BamTools index format

Indexing a sorted BAM file allows a (semi-)random-access jump to a particular region of interest. The BAM format describes a standard index format (BAI), which uses a binning scheme similar to the one implemented in the UCSC Genome Browser (Kent *et al.*, 2002). This BAI scheme provides quick access to the beginning of the contiguous run of alignments (or 'chunk') that overlaps the beginning of a region of interest. However in large datasets, where coverage is high, a significant number of alignments may be read and immediately discarded before finding an alignment that actually overlaps the region of interest. In response to this, we created an alternative BamTools index scheme (BTI) that is based, not on alignment position, but on a fixed read count. Thus, the number of alignments that must be read and discarded always has a fixed upper bound. Choosing the bin size depends on a trade-off between disk space versus access speed, and can be configured as needed. In our own index files, we use bins of 1000 alignments. Based on 1 million jumps to random positions in the chromosome 20 BAM file mentioned above, our new indexing method provides, on average, 10 times faster access than the standard BAI (with an SD roughly 20 times lower). The slowest access time (worst case) was also roughly 10 times faster for BTI than the worst case for BAI (Table 2).

**Table 2.** BamTools index performance

| Index type | Jump times (μs) | | |
|---|---|---|---|
| | Mean | SD | Worst-case |
| BAI | 446 | 13669.81 | 1 291 278 |
| BTI | 40 | 665.38 | 121 403 |

Generated from performing 1 million random jumps in a BAM file containing 116-fold coverage of human chromosome 20.

## 3 CONCLUSION

The BamTools C++ API/library has been successfully integrated into a variety of applications. It provides the BAM file support for several utilities in the BEDtools suite (Quinlan *et al.*, 2010). Using BamTools, the freeBayes variant caller (https://github.com/ekg/freebayes) has produced whole-genome calls for the NCBI's new 1000 Genomes pipeline on up to 1000 BAM files simultaneously. The data management utilities (indexing, merging, etc.) provided by the BamTools command-line toolkit also play an integral role in this pipeline. Visualization software like Gambit (http://bioinformatics.bc.edu/marthlab/Gambit), uses the BamTools API to efficiently access alignment data in a setting where real-time performance is critical.

## ACKNOWLEDGEMENT

We would like to thank Baptiste Lepilleur for the public-domain JSON parser (JsonCPP) and Heng Li, primary developer of the SAMtools suite.

*Funding*: National Institutes of Health grants (RO1 HG004719 and RC2 HG005552).

*Conflict of Interest*: none declared.

## REFERENCES

The 1000 Genomes Project Consortium. (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.

Kent,W.J. *et al*. (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1106.

Li,H. *et al*. (2009) The sequence alignment/map format and SAMtools. *Bionformatics.*, **25**, 2078–2079.

McKenna, A. *et al*. (2010) The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, **20**, 1297–303.

Quinlan,A.R. *et al*. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bionformatics.*, **26**, 841–842.