

Samtools in Java

Formats and tools for managing next-
gen sequencing data

<http://samtools.sourceforge.net/SAM-1.3.pdf>

Outline

- Review
 - Illumina sequencing, FASTQ, Alignments
- SAM and BAM
 - Compact alignment formats for NGS
- Samtools
 - Unix utilities for working with SAM/BAM files, NGS

<http://samtools.sourceforge.net/SAM-1.3.pdf>

Illumina sequencing

- Create a “library”
 - From cDNA (RNA-Seq) or genomic DNA
 - Depends on application
 - Sequence by synthesis on Illumina sequencer
 - One flow cell generates tens of millions of “reads”
 - Output is “FASTQ”
 - Sequence read name, quality score, sequence itself
- Align reads onto reference sequence
 - Stored in “SAM” format

What is a SAM?

- SAM - Sequence Alignment/Map format
- Tab-delimited text format with two sections:
 - Header – meta-data about the alignments
 - Alignments themselves
 - Each alignment is on one line, has 11 required fields, many optional fields that use an easy-to-parse syntax
- SAM files sorted by start position
 - Makes searching faster

<http://samtools.sourceforge.net/SAM-1.3.pdf>

BAM is binary SAM

- BAM uses block compression scheme to make alignments more compact
- BAM files can be sorted and indexed
 - Index contains locations of blocks in the larger file
 - Makes accessing data very fast
- BAI (extension .bai) is the index for a BAM file
 - Sample.bam has index file Sample.bam.bai

<http://samtools.sourceforge.net/SAM-1.3.pdf>

Use samtools (command line tools) to view contents of BAM

- samtools
 - Unix-friendly command line program for viewing contents of BAM file
 - Open source project
 - Started mainly to support 1000 genomes project
- 6111 (Week 15) Web site has a link to “zip” file containing samtools compiled for Mac
- See lectures, notes from 6111
 - <http://transvar.org/6111>

Header

- Each line starts with @ and is tab-delimited
 - @HD comes first, followed by version string indicating the SAM file version and SO tag indicating whether the file is sorted
 - @SQ comes next (if file is sorted in any way)
 - This is the “sequence dictionary” and reports the names and lengths of the reference sequences used to make the alignments

View just the header

```
$ samtools view -H treatment_drought2_out.bam
@HD VN:1.0 SO:sorted
@SQ SN:chr1 LN:30427671
@SQ SN:chr2 LN:19698289
@SQ SN:chr3 LN:23459830
@SQ SN:chr4 LN:18585056
@SQ SN:chr5 LN:26975502
@SQ SN:chrC LN:154478
@SQ SN:chrM LN:366924
@PG ID:TopHat VN:1.1.4 CL:/common/lorainelab/sw/
tophat-1.1.4.Linux_x86_64/tophat --max-multihits 1 --min-
intron-length 20 --solexa1.3-quals -I 1200 -p 8 -F 0 -o
tophat-1.1.4-pilot-pollen/treatment_drought2_out -G /storage/
lorainelab/fastapub/tair/TAIR9_GFF3_genes-fixed.gff /storage/
lorainelab/bowtieindex/a_thaliana chapel_hill_1/
s_8_sequence.txt,chapel_hill_2/s_4_sequence.txt
```

Alignments

1.4 The alignment section: mandatory fields

Each alignment line has 11 mandatory fields. These fields always appear in the same order and must be present, but their values can be '0' or '*' (depending on the field) if the corresponding information is unavailable. The following table gives an overview of the mandatory fields in the SAM format:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	Query template NAME
2	FLAG	Int	[0,2 ¹⁶ -1]	bitwise FLAG
3	RNAME	String	* [!-()+-<>-~] [!-~]*	Reference sequence NAME
4	POS	Int	[0,2 ²⁹ -1]	1-based leftmost mapping POSition
5	MAPQ	Int	[0,2 ⁸ -1]	MAPping Quality
6	CIGAR	String	* ([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	* = [!-()+-<>-~] [!-~]*	Ref. name of the mate/next fragment
8	PNEXT	Int	[0,2 ²⁹ -1]	Position of the mate/next fragment
9	TLEN	Int	[-2 ²⁹ +1,2 ²⁹ -1]	observed Template LENgth
10	SEQ	String	* [A-Za-z=.]+	fragment SEQUENCE
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

1. QNAME: Query template NAME. Each template has a unique name.

Use samtools view to retrieve reads overlapping a region of interest

```
$ samtools view treatment_drought2_out.bam
chr1:1,000,000-1,000,001 | more
61C2DAAXX:4:91:1662:10658#0      16      chr1      999935
255      75M      *      0      0
TTAAGGCTCCCATTTACACTATCGAAAAAGATGGGACAAGTGCTGAAACGTGTATGAT
CCGGTTCCATGCTGGTC
4BBBBBCBBBACBBBBCBCBCDC@BCCCC@CBCACBCCCCCCCCCCCCBCCCCCCCCCCC
CCCCCCCCCCCCCCCC      NM:i:0  NH:i:1
```

Mandatory Fields

- | | |
|---|---|
| 1 – QNAME name of the query sequence | 6 – CIGAR describes the alignment |
| 2 – FLAG bitwise flag score (ignore for now) | 7 – RNEXT (ignore for now) |
| 3 – RNAME name of the reference sequence | 8 – PNEXT (ignore for now) |
| 4 – POS start position of the alignment (one-based) | 9 – TLEN (ignore for now) |
| 5 – MAPQ alignment quality score (ignore for now) | 10 – SEQ query sequence |
| | 11 – QUAL base quality (ignore for now) |

Optional fields

- Most alignment tools include at least some in their output

- Syntax

- TAG:TYPE:VALUE

- Example

- NM:i:0

- NH:i:1

NM – edit distance to the reference
(number of bases that need to change to perfectly match the reference)

NH – Number of reported alignments that contain the query in the current record – the number of times the read aligned

i – type is integer

For more, see the SAM spec

CIGAR

- Use this to calculate the end position of the alignment
- Example
 - Read X starts at position 100 and has CIGAR code 75M (75 matches)
 - Start position in interbase is $100-1=99$
 - End position is $99+75=74$

Java samtools

- See package:
 - net.sf.samtools
- To open and read a BAM file create an instance of:
 - net.sf.samtools.SAMFileReader
 - See javadocs:
 - <http://picard.sourceforge.net/javadoc/net/sf/samtools/SAMFileReader.html>

Constructor Summary

SAMFileReader(File file)

Prepare to read a SAM or BAM file.

SAMFileReader(File file, boolean eagerDecode)

Read a SAM or BAM file, possibly with an index file if present.

SAMFileReader(File file, File indexFile)

Prepare to read a SAM or BAM file.

SAMFileReader(File file, File indexFile, boolean eagerDecode)

Read a SAM or BAM file, possibly with an index file.

SAMFileReader(InputStream stream)

Prepare to read a SAM or BAM file.

SAMFileReader(InputStream stream, boolean eagerDecode)

Read a SAM or BAM file.

SAMFileReader(SeekableStream strm, File indexFile, boolean eagerDecode)

Read a BAM file via caller-supplied mechanism.

SAMFileReader(URL url, File indexFile, boolean eagerDecode)

Read a BAM file by http indexed query will be allowed.

eagerDecode - If true, all SAMRecords are fully decoded as they are read
Set to "false" if all we want to do is count and we don't need any further information

<http://picard.sourceforge.net/javadoc/net/sf/samtools/SAMFileReader.html>

query

```
public SAMRecordIterator query(String sequence,  
                                int start,  
                                int end,  
                                boolean contained)
```

Iterate over records that match the given interval. Only valid to call this if `hasIndex() == true`.

Only a single open iterator on a given `SAMFileReader` may be extant at any one time. If you want to start a second iteration, the first one must be closed first. You can use a second `SAMFileReader` to iterate in parallel over the same underlying file.

Note that indexed lookup is not perfectly efficient in terms of disk I/O. I.e. some `SAMRecords` may be read and then discarded because they do not match the interval of interest.

Note that an unmapped read will be returned by this call if it has a coordinate for the purpose of sorting that is in the query region.

Parameters:

`sequence` - Reference sequence of interest.

`start` - 1-based, inclusive start of interval of interest. Zero implies start of the reference sequence.

`end` - 1-based, inclusive end of interval of interest. Zero implies end of the reference sequence.

`contained` - If true, each `SAMRecord` returned is will have its alignment completely contained in the interval of interest. If false, the alignment of the returned `SAMRecords` need only overlap the interval of interest.

Returns:

Iterator over the `SAMRecords` matching the interval.

<http://picard.sourceforge.net/javadoc/net/sf/samtools/SAMFileReader.html>

Simple examples – in “class”

- Update your copy of the repository in the “class” folder
 - Get project SimpleBamReaderExample
 - This has today’s code examples
 - Note: You can use this as a starting point for your homework, but *you must improve it*
 - Required for full credit
 - More on this later...
 - Get new folder “class/lib”
 - Contains JAR files you’ll need to access BAM files in Java

To use samtools, picard in your programs:

- Tell NetBeans where to find net.sf.samtools and related packages:
- Add “jar” files to your project’s build path
 - Right-click your project
 - Choose Properties > Libraries > Add JAR/Folder
 - Select class/lib/picard.jar (after updating)
 - Repeat for class/lib/sam.jar
- Now your program can access the picard/samtools packages

Download sample files

- Go to <http://igbquickload.org/samples>
- Control-click “bam” and “bai” files
 - Choose ‘Save As’
 - Download onto your computer
- Don’t just click them
 - You can’t view them in your Web browser
- Or download in the Terminal using “curl -O”
 - curl -O [URL]

Main.java

- Shows how to open a file
- Create a SAMFileReader that can read the file
- Query the file and count reads

JFileChooserDemo

- Shows how to create a JFileChooser
- Use it to select BAM and BAI files
- Read a file and count reads

Improve on the example code in your project (**required!**)

- Handle user input properly
 - What if users select a non-BAM file?
 - What if users select more than one BAM file?
 - What if the BAI file is not available?
- Make life easier for the user
 - How can you get your program to “remember” previously-selected folders?
 - Hint: re-use the JFileChooser
 - Index files usually have the same name as the “bam” file, but with “.bai” added. Use this to save the user the hassle of clicking both “bai” and “bam” file when s/he only needs to choose one.