SAM BAM CRAM Mapped sequence data formats

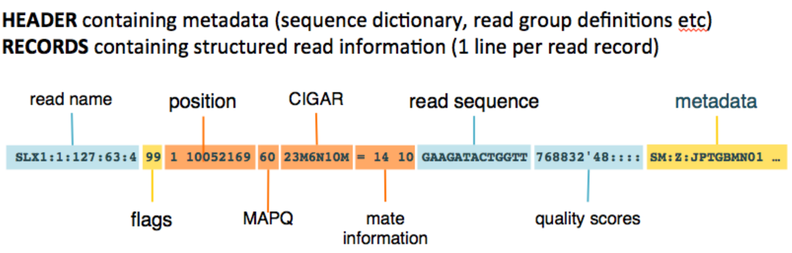
IMPORTANT: This is the legacy GATK Forum website. This information is only valid until Dec 31st 2019. For latest documentation and forum click here

created by GATK_Team
on 2017-12-24

SAM, BAM and CRAM are all different forms of the original SAM format that was defined for holding aligned (or more properly, *mapped*) high-throughput sequencing data. SAM stands for Sequence Alignment Map and is described in the standard specification here. BAM and CRAM are both compressed forms of SAM; BAM (for Binary Alignment Map) is a lossless compression while CRAM can range from lossless to lossy depending on how much compression you want to achieve (up to *very much indeed*). BAMs and CRAMs hold the same information as their SAM equivalent, structured in the same way; what is different between them is how the files themselves are encoded.

Most genomes mappers (including BWA and STAR, our favorite mappers for DNA and RNAseq, respectively) emit SAM or BAM natively. Note that there is a variant of BAM that we call uBAM (for *unmapped* BAM) which holds unmapped read data. You can learn more about our "off-label" use of this format here.

The basic structure of the SAM format is depicted in the figure below:



GATK and Picard requirements

All GATK tools that take in mapped read data expect a BAM file as primary format. Some support the CRAM format, but we have observed performance issues when working directly from CRAM files, so in our own work we convert CRAM to BAM first, and we only use CRAM for archival purposes. Reading SAM files directly is not supported by current GATK tools, but they can easily be converted with Picard tools, which are conveniently bundled into GATK4. In most of our documentation we only feature BAM because it is the most commonly used form used for analysis, even if the other formats are supported as well. See the Tool Docs for tool-specific format requirements.

A few additional requirements:

- SAM files must be named with the `.sam` extension, BAM with `.bam`, and CRAM with `.cram`.
- All SAM, BAM or CRAM files must be accompanied by an index, which can be generated by Picard BuildBamIndex.
- All SAM, BAM or CRAM files that you want to analyze should pass validation by ValidateSamFile from Picard.
- The header should contain read groups with sample names.
- Every read in a BAM must belong to a read group listed in the header.
- GATK tools require that the reads be sorted in coordinate order (not by queryname and not "unsorted"). Some Picard tools are able to work with other sort orders.
- GATK doesn't impose a particular ordering scheme for contigs, but it does expect contigs to be ordered the same way across files (including FASTA and VCF files). Everything should match the sequence dictionary of the reference genome FASTA.

Aside from file validation, which is its own thing, the easiest way to check whether a BAM file complies with these requirements it is to run the following Samtools command to view its header:

```
samtools view -H read_data.bam
```

This produces the following output (truncated for space):

```
@HD VN:1.5  GO:none SO:coordinate
@SQ SN:20   LN:63025520 UR:http://www.broadinstitute.org/ftp/pub/seq/references/ref.fasta   M5:0dec9660ec1efaaf33281c0d5ea2560
@RG ID:H0164.2  PL:illumina PU:H0164ALXX140820.2    LB:Solexa-272222    PI:0    DT:2014-08-20T00:00:00-0400 SM:NA12878  CN:BI
@PG [program group metadata]
```

More about read groups

The presence of the `@RG` tags indicate the presence of read groups. Each read group has a `SM` tag, indicating the sample from which the reads belonging to that read group originate.

In addition to the presence of a read group in the header, each read must belong to one and only one read group. You can look at reads that have an RG tag by running the following command:

```
samtools view read_data.bam | grep 'RG:Z' | less
```

This will show records like this one:

```
H0164ALXX140820:2:1222:11840:58673      99      20      9999905 60      92M2I57M        =       10000151        397         TAA
```

The read group that this read belongs to is specified by the `RG:Z:H0164.2` tag, which tells us it belongs to read group H0164.2. Looking that up in the header lines above, we find that the read belongs to the sample with the SM tag NA12878.

For even more about read groups and how to define them, see the Dictionary entry here.

More about sort order

The `SO:coordinate` flag tells you what is the ordering of reads in your BAM file. If you have more than one contig in your file (which is usually the case; but here we're working with a tiny demo file) you'll have to infer the contig order by, you know, looking at it.

There's a Picard tool called SortSam that will sort a BAM file according to a given sort order parameter. A similar utility exists in Samtools, but we recommend the Picard tool because SortSam will also set a flag in the header that specifies that the file is correctly sorted, and this flag is necessary for the GATK to know it is safe to process the data. For contigs, there's also a tool in Picard called ReorderSam that allows you to reorder the contigs to match a given reference sequence dictionary.

Subsetting read data from a BAM

There are good ways, mediocre ways, and downright disastrously wrong ways to subset read data. The biggest pitfall that you must absolutely avoid is messing with the sequence dictionary. It may seem like a good idea to reduce the file size by isolating a single chromosome -- but unless you're doing it strictly for testing or demo purposes (like we do with our mini-reference, shown above), chances are you're shooting yourself in the foot and will come to regret it.

If you don't believe us, take it from Sam Nicholls, a GATK user who went through a very painful experience and wrote it up in painstaking (and quite humorous) detail in his blog.

TL;DR: You can use intervals to restrict any analysis to a subset of your data, or use PrintReads (again with intervals) to subset/extract data to a separate file.

Updated on 2017-12-26