Samtools

Contents

Introduction

Samtools hardly needs an introduction, it is one of the cornerstones of bioinformatics processing and is at the heart of the business of sequence mapping / aligning.

Despite that introduction, note that samtools does not actually carry out alignments itself. Rather it offers utilities attendant on real aligners such as **bwa** and **bowtie**.

Primarily this is due to its providing various tools (available as subcommands) centred on a well defined sequence alignment format, **sam**, and its binary (and therefore compressed) equivalent, **bam**.

This wiki page just offers some tips on how to use it, as there is plenty documentation elsewhere, some of which is mentioned in the links.

Please note that around 2014, the samtools code base went through a restructuring in order to expose more of its intermediate steps, the result of which was the HTS library. Many other tools which use samtools do not use this so the default version of samtools on the marvin cluster is 0.1.19 which is in fact the older generation of samtools code.

Version 0.1.19

This consists of the following tools

- ace2sam
- bamcheck
- maq2sam-long
- maq2sam-short
- md5fa
- md5sum-lite
- samtools
- wgsim
- tabix
- bgzip
- bcftools
- vcfutils.pl

General Tips

1. samtools commands follow the subcommand style: i.e. they always start with a "samtools" followed by subcommand (eg. "sort", "view", etc) which identify the exact operation samtools will perform. This is due to the fact that samtools is a suite of tools for handling alignment files in sam/bam format.
2. Two input files will commonly be needed, often a sam/bam file and also the reference file.
3. **view** is the commonly used subcommand, which itself has a number of extra option and the name however might refer to internal viewing, because external user visuals is not samtools strong point, though it has some capabilities in this regard ("tview").
4. Beware the three broad versions of samtools. Those beginning with major version number 0, as in 0.1.19 are the old samtools. Those beginning with major version number 1, as in 1.2, were the new generation, whose internal structre changed though the subcommands were similar. Finally the 1.3 version introduced a wider range of functions. In many ways this third version should have been major version number 2, but that didn't happen.

old samtools vs. new samtools

Here is a list of the old version 0 samtools commands with examples:

```
samtools view -bt ref_list.txt -o aln.bam aln.sam.gz
samtools sort aln.bam aln.sorted
samtools index aln.sorted.bam
samtools idxstats aln.sorted.bam
samtools merge out.bam in1.bam in2.bam in3.bam
samtools faidx ref.fasta
samtools pileup -vcf ref.fasta aln.sorted.bam
samtools mpileup -C50 -gf ref.fasta -r chr3:1,000-2,000 in1.bam in2.bam
samtools tview aln.sorted.bam ref.fasta
bcftools index in.bcf
```

Here is a list of those available in the third generation version 1.3, together with examples:

```
samtools view -bt ref_list.txt -o aln.bam aln.sam.gz
samtools sort -T /tmp/aln.sorted -o aln.sorted.bam aln.bam
samtools index aln.sorted.bam
samtools idxstats aln.sorted.bam
samtools flagstat aln.sorted.bam
samtools stats aln.sorted.bam
samtools bedcov aln.sorted.bam
samtools depth aln.sorted.bam
samtools view aln.sorted.bam chr2:20,100,000-20,200,000
```

```
samtools merge out.bam in1.bam in2.bam in3.bam
samtools faidx ref.fasta
samtools tview aln.sorted.bam ref.fasta
samtools split merged.bam
samtools quickcheck in1.bam in2.cram
samtools dict -a GRCh38 -s "Homo sapiens" ref.fasta
samtools fixmate in.namesorted.sam out.bam
samtools mpileup -C50 -gf ref.fasta -r chr3:1,000-2,000 in1.bam in2.bam
samtools flags PAIRED,UNMAP,MUNMAP
samtools fastq input.bam > output.fastq
samtools fasta input.bam > output.fasta
samtools addreplacerg -r 'ID:fish' -r 'LB:1334' -r 'SM:alpha' -o output.bam input.bam
samtools collate aln.sorted.bam aln.name_collated.bam
samtools depad input.bam
```

Common commands

A **bam** file is the compressed binary version of a **sam** file. It exists purely for efficiency purposes and contains the same information. Because it is not human-readable, converting out of, but also back into bam, is very frequent activity. It is done via the **view** subcommand in the following canonical way of converting from sam to bam, and it needs to be redirected into bam filename:

```
samtools view -S -h -b input_sam_filename > your_chosen_bam_filename
```

Explanation:

- **-S**, this says the input is SAM, though in the latest versions of samtools this is the default and so may be left out. From the manual: "-S: Ignored for compatibility with previous samtools versions. Previously this option was required if input was in SAM format, but now the correct format is automatically detected by examining the first few characters of input." However it's best to keep it as one can never be too sure of which version of samtools is running.
- **-b**, refers to the output being BAM.
- **-o**, does not refer to "output name" like it usually might, it refers to STDOUT. This is for piping convenience.
- **-h**, is for making sure the header is passed to the bam. Without this option, the default of "noheader" is in force.
- note that the bam extension will **not** be added automatically. You need to give the name plus extension after the -o switch. Note that the **sort** subcommand does do this however.

Bamfiles are unique to the reference they're aligned against, though it is not often clear which version of the reference was used. This may be available in the bamfile's header information, so you can try and use

```
samtools view -H
```

to see if you can find the right reference file.

Once you have a bam file, it's usual to quickly proceed to sorting and then indexing (for which sorting is necessary) because many tools that will typically be run afterwards, require it. Sorting is often as easy as the following, and this is typically the command-line you'll need if you're on **version 0.1.19** or earlier:

```
samtools sort aln.bam aln_sorted
```

Note here how the sot subcommand automatically add a **bam** extension to the second argument. In **versions 1.x.x** i.e. more modern versions, of samtools, you need the -o, as in:

```
samtools sort aln.bam -o aln_sorted.bam
```

Getting these two mixed up, especially running the later on a earlier version will cause garbled screen syndrome, which can only be cured by terminating.

Indexing is quite quick, and also seems to generate the index file (which appends a .bai extension) in the subdirectory where the bams are found. So, the following command:

```
for i in bams/*.bam;do samtools index $i;done
```

will generate the index files in the ./bam subdirectory, despite the fact that no output directory was specified.

Note that the index subcommand is fine for bam files, but that when we want to index fasta files, we should use the faidx subcommand, as in

```
samtools faidx chimpHg19.fa.gz
```

which will append the fai extension to the fa.gz file.

When **view** is used, it outputs headerless apparently. You can stop this with the **-h**.

Counting mapped reads

A major indicator of how an alignment has fared is by counting the mapped vs. unmapped reads. The FLAG field in the sam format is used for this.

```
samtools view -b -F 4 -c ERR131815.bam
```

A flag field of 4 with a capital F, and the -c option will specify counting. A lower case f will count unmapped reads.

tview

While samtools does not prioritise visualisation, it is still abel to perform it. The tview subcommand does provide a raw view of the alignment that a bam files has. Sometimes such raw, and less pretty representations of the alignment can be useful.

It requires the bam file, which must be indexed and (often, sorted) and the reference sequence, and it is runs like so:

```
samtools tview alignments/sim_reads_aligned.sorted.bam genomes/NC_008253.fna
```

As Heng Li, the developer of samtools is an avid **vi** fan, the keybinding all reflect vi's keys so that **I** move you

Explanation:

- **NC_008253.fna** is the reference file here, the original read file is not needed.
-

- You can see here the point relating to tip no .2 above ... the two arguments are the bam file (first argument) and its associated reference (second argument) with no option switches.

mpileup

This is the subcommand to use to call the variants, most often - but not always - SNPs. The output is formatted in the **VCF** format, or its compressed binary equivalent, **BCF**. Here is an example of calling it:

```
samtools mpileup -u -f samtools_bowtie/NC_012967.1.fasta samtools_bowtie/SRR030257.sorted.bam > samtools_bowtie/SRR030257.bcf
```

Explanation:

- Note how the reference and the bam file and the directed output all go to a subdirectory
- -u is ideal for a piped command such as this one, because it's for uncompressed output.
- With no options determining the output format, the output will be **BCF**

To visualise, the output needs to be converted to text-based VCF with the following command:

```
bcftools view -v -c -g samtools_bowtie/SRR030257.bcf > samtools_bowtie/SRR030257.vcf
```

merge

Samtools does indeed have a merge subcommand but it is unpopular. Practitioners generally use picard tools instead. This has to do with **merge** not establishing a read-grup or @RG for each component bam file.

Functionality relating to SAM format

This should include all possible functionality, as this is samtools' software brief, to be able to thandle the file format. Because of this also, many samtools' options require knowledge of the file format. Here are a few of the less obvious ones.

FLAG related options

There is just one flag which sets various bits as 0 or 1 according to certain properties.

- **-f** and **-F**. These are opposites of each other and refer to filtering on certain bits in the FLAG field. Lower case f filters out everything but those sequences with the appropriate FLAG bit set. And so the uppercase F filters just those sequences with the bits in the flag set.

Links

- Ethan Cerami's Primer
- samtools' manual page
- meaning of the FLAG value from samformat.info

Installation Notes

Since version 1.2, samtools build structure has changed. Before it was monolithic, since 1.2 it is split into a library, htslib and smatools proper. A third package, bcftools, is often mentioned in the same breath, though