Samtools: viewing, counting and sorting your alignment data

- Introduction to Samtools - manipulating and filtering bam files
- Sorting and Indexing a bam file: samtools index, sort
- Samtools flags and mapping rate: calculating the proportion of mapped reads in an aligned bam file
- Filtering bam files based on mapped status and mapping quality using samtools view

Introduction to Samtools - manipulating and filtering bam files

As we showed you yesterday, the main type of output from aligning reads to a databases is a binary alignment file, or BAM file. These files are compressed, so they can't be viewed using standard unix file viewers such as more, less and head. **Samtools** allows you to manipulate the **.bam** files - they can be converted into a non-binary format (SAM format specification here) and can also be ordered and sorted based on the quality of the alignment. This is a good way to remove low quality reads, or make a BAM file restricted to a single chromosome.

We'll be focusing on just a few of **samtools** functions in this series of exercises. Since most aligners produce a BAM file, we'll work on some basic manipulations of the BAM files we produced from our alignments yesterday.

Most functionality while using BAM files can be described as such:

1. SAM files are converted into BAM files (**samtools view**)
2. BAM files are sorted by reference coordinates (**samtools sort**)
3. Sorted BAM files are indexed (**samtools index**)
4. Sorted, indexed BAM files are ***filtered*** based on location, flags, mapping quality (**samtools view** with filtering options)

Take a look here for a detailed manual page for each function in **samtools**. These steps presume that you are using a mapper/aligners such as **bwa**, which records both mapped and unmapped reads - make sure you check how the aligner writes it's output to SAM/BAM format, or you may get a strange surprise in your output aligned files!

The code block below details some basic **samtools** functionality:

basic samtools functionality

```
samtools view -o outfile_view.bam infile.bam  # use the -c option to just count alignment records
samtools sort infile.bam outfile.sorted.bam
samtools index aln.sorted.bam
```

First, logon to **stampede** and copy the file **yeast_pairedend.bam** to your $SCRATCH directory:

copy the yeast_pairedend.bam file from yesterday to a new directory "samtools"

```
ssh user@login8.stampede.tacc.utexas.edu
cd $SCRATCH/core_ngs
mkdir samtools
cd samtools
cp  $SCRATCH/core_ngs/alignment/yeast_pairedend.bam .
```

Sorting and Indexing a bam file: samtools index, sort

Now that we have a BAM file, we need to index it. All BAM files need an index, as they tend to be large and the index allows us to perform computationally complex operations on these files without it taking days to complete.

Exercise 1: Sort and index the file "yeast_pairedend.bam", then examine the files you created

solution

```
module load samtools
samtools sort yeast_pairedend.bam yeast_pairedend_sort # will take 1-2 minutes
samtools index yeast_pairedend_sort.bam
```

Use **ls -lah** to see what files you made and how large they are. This is what mine look like:

```
-rwxrwxr-x 1 awh394 G-801021 110M May 26 14:12 yeast_pairedend.bam
-rwxrwxrwx 1 awh394 G-801021  91M May 26 14:13 yeast_pairedend_sort.bam
-rwxrwxrwx 1 awh394 G-801021  20K May 26 14:14 yeast_pairedend_sort.bam.bai
```

Note how small the index file is!

Samtools flags and mapping rate: calculating the proportion of mapped reads in an aligned bam file

We have a sorted, indexed BAM file. Now we can use other **samtools** functionality to filter this file and count mapped vs unmapped reads in a given region. **samtools** allows you to sort based on certain flags that are specified on page 4 on the sam format specification. We'll focus on a couple, below.

Bit	Description	
0x1	template having multiple segments in sequencing	*1 = part of a read pair*
0x2	each segment properly aligned according to the aligner	*1 = "properly" paired*
0x4	segment unmapped	*1 = read did **not** map*
0x8	next segment in the template unmapped	*1 = mate did **not** map*
0x10	SEQ being reverse complemented	*1 = minus strand read*
0x20	SEQ of the next segment in the template being reversed	*1 = mate on minus strand*
0x40	the first segment in the template	*1 = R1 read*
0x80	the last segment in the template	*1 = R2 read*
0x100	secondary alignment	*1 = secondary possible hit*
0x200	not passing quality controls	
0x400	PCR or optical duplicate	*1 = marked as duplicate*

Here are three of the most useful flags to sort on. We'll be using the ***unmapped*** flag.

a few samtools flags

```
# SAM specifications common flag usage
0x04 = unmapped
0x02 = part of a properly aligned pair
0x400 = optical duplicate  # look at samtools rmdup if you need to remove these sequences
```

Exercise 2: count unmapped reads vs total reads on chromosome III for the yeast_pairedend_sort.bam file you created above. What proportion of the reads are mapped?

I've put my output for each line in the comment area.

solution

```
module load samtools                                    # if needed
samtools view -c yeast_pairedend_sort.bam chrIII        # total number of reads on this chromosome: 15503
samtools view -c -F 0X04 yeast_pairedend_sort.bam chrIII # reads on this chromosome which are unmapped: 13973
```

So the total proportion of reads that were unmapped on chromosome III is 13973/15503 or 90.1%, which is really high! Only ~10% of reads on this chromosome were able to be mapped to the genome.

Filtering bam files based on mapped status and mapping quality using samtools view

Mapping qualities are a measure of how likely a given sequence alignment to a location is correct. The lowest score is a mapping quality of zero, or **mq0** for short. The reads map to multiple places on the genome, and we can't be sure of where the reads originated. To improve the quality of our data, we can remove these low quality reads from our sorted and indexed file.

Exercise 3: Remove unmapped and low quality reads from your bam file.

solution

```
module load samtools # if needed
samtools view -b -F 0x04 -q 1 -o yeast_pairedend_sort.mapped.q1.bam yeast_pairedend_sort.bam
```

Here is what my output looks like when I use **ls -lah** after running the above commands:

samtools filtering output

```
-rwxrwxrwx 1 awh394 G-801021  42M May 26 14:26 yeast_pairedend_sort.mapped.q1.bam
```

Now that we have a BAM file with only high-quality mapped reads, we are ready to manipulate this file using **bedtools**.