2013 - BMMB 597D: Analyzing Next Generation Sequencing Data

Week 8, Lecture 16

István Albert

Biochemistry and Molecular Biology
and Bioinformatics Consulting Center

Penn State

Binary SAM (BAM) files

SAM file:

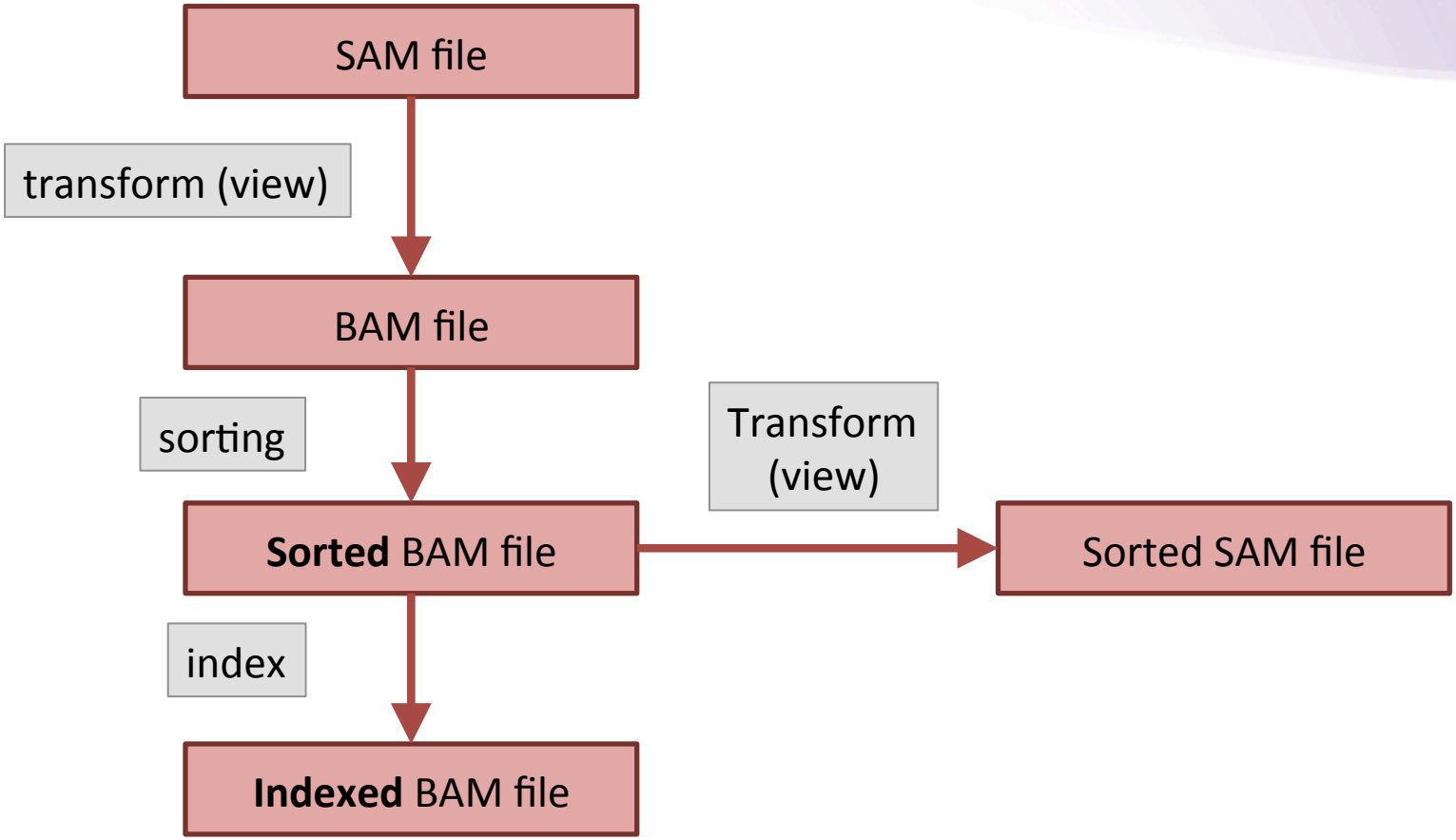
- information on the alignment of each read
- optimized for readability and sequential access

BAM (binary SAM):

- compression → saves space (optimized for size)
- may be **sorted** + **indexed** → location query (optimized for random access)
- the file is not readable by eye

Your default format should be BAM – only turn it into SAM when viewing the file

SAM/BAM hierarchy



Some tools have certain requirements of what type of SAM/BAM they take.

Your default data format should be a **sorted, indexed BAM** file!

Download and 'make' SAMTOOLS

SAMtools

SOURCEFORGE.NET®

Home

Introduction

SAM (Sequence Alignment/Map) format is a generic format for storing large nucleotide sequence alignments. SAM aims to be a format that:

- Is flexible enough to store all the alignment information generated by various alignment programs;
- Is simple enough to be easily generated by alignment programs or converted from existing alignment formats;

General Information

SAM Spec v1.4
SF Project Page
SF Download Page
Mailing Lists
SVN Browse
Related Software
FAQ

http://samtools.sourceforge.net/

Samtools: is suite of commands

```
Usage:   samtools <command> [options]

Command: view        SAM<->BAM conversion
         sort        sort alignment file
         mpileup     multi-way pileup
         depth       compute the depth
         faidx       index/extract FASTA
         tview       text alignment viewer
         index       index alignment
         idxstats    BAM index stats (r595 or later)
         fixmate     fix mate information
         flagstat    simple stats
         calmd       recalculate MD/NM tags and '=' bases
         merge       merge sorted alignments
         rmdup       remove PCR duplicates
         reheader    replace BAM header
         cat         concatenate BAMs
         bedcov      read depth per BED region
         targetcut   cut fosmid regions (for fosmid pool only)
         phase       phase heterozygotes
         bamshuf     shuffle and group alignments by name
```

Most actions will provide help on their usage

```
$ samtools view

Usage:   samtools view [options] <in.bam>|<in.sam> [region1 [...]]

Options: -b       output BAM
         -h       print header for the SAM output
         -H       print header only (no alignments)
         -S       input is SAM
         -u       uncompressed BAM output (force -b)
         -1       fast compression (force -b)
         -x       output FLAG in HEX (samtools-C specific)
         -X       output FLAG in string (samtools-C specific)
         -c       print only the count of matching records
         -B       collapse the backward CIGAR operation
         -@ INT   number of BAM compression threads [0]
         -L FILE  output alignments overlapping the input BED FILE [null]
         -t FILE  list of reference names and lengths (force -S) [null]
         -T FILE  reference sequence file (force -S) [null]
         -o FILE  output file name [stdout]
         -R FILE  list of read groups to be outputted [null]
         -f INT   required flag, 0 for unset [0]
```

Default Operation

- By default **samtools** expects a **BAM** file as input and will produce a **SAM** file as output

- Every alignment result should be stored as a **sorted** and **indexed** BAM file

Transform SAM to BAM

transform to bam

```
samtools view –Sb input.sam > tempfile.bam
```

sort bam file

```
samtools sort –f tempfile.bam output.bam
```

Index bam file

```
samtools index output.bam
```

Add the following to the previous week's shell script

```
# perform the alignments via bwa
~/bin/bwa aln $REF $QUERY > $SAI
~/bin/bwa samse $REF $SAI $QUERY > $SAM

# transform the SAM file to BAM
~/bin/samtools view -Sb $SAM > $TMP

# sort the samfile
~/bin/samtools sort -f $TMP $BAM

# index the BAM file
~/bin/samtools index $BAM

echo "Finished ref=$REF, query=$QUERY, bam=$BAM"
```

Filtering SAM/BAM files

Required flag (keep if matches)

```
samtools view -f
```

Filtering flag (remove if matches)

```
samtools view -F
```

Flags are using a bitwise representation

1 = 00000001 → paired end read
2 = 00000010 → mapped as proper pair
4 = 00000100 → unmappable read
8 = 00001000 → read mate unmapped
16 = 00010000 → read mapped on reverse strand

```
ialbert@porthos ~/work/lec12
$ ~/bin/samtools view -c -f 4 results.bam
1

ialbert@porthos ~/work/lec12
$ ~/bin/samtools view -c -F 4 results.bam
3
```

-c means to count the lines
-f <number> - keep reads that match
-F <number> - remove reads that match

```bash
# save on typing
alias samtools=~/bin/samtools

# how many reads in total
samtools view -c results.bam

# reads that cannot be mapped
samtools view -c -f 4 results.bam

# reads that can be mapped
samtools view -c -F 4 results.bam

# reads that map to reverse strand
samtools view -c -f 16 results.bam

# reads that map to forward strand
samtools view -c -F 16 results.bam

# reads that have a minimum mapping quality of 1
# note that for BWA this also means unique alignment!
samtools view -c -q 1 results.bam
```

Ready Mac-Roman Ln: 2 Col: 1 Bash

A sorted file will stay sorted during transformation

- Once sorted all output will stay sorted regardless of the output type (SAM, BAM)
- You can creating a second, smaller and filtered file that does not need to be sorted again.
- You do need to index the new file though!

Explore other commands

Flag statistics

```
samtools flagstat data.bam
```

Index stats

```
samtools idxstats data.bam
```

Depth of coverage

```
samtools depth data.bam | head
```

Querying a BAM file **name:start-end**

Samtools allows querying:

```
samtools view data.bam  chrV:1000-2000
```

Homework 16

Generate a **sorted** and **indexed** BAM file based on the data **lect15.fq.gz**

1. Find the number of uniquely mapped reads

2. Find the number of high quality alignments (MAPQ>30) for each strand separately

3. A genomic feature has its start site on the forward strand on chromosome I at position 111,000.

   – How many reads fall within 500b upstream of this location?
   – Print the position of each read (hint: there are not that many)
   – Report the number of reads in this region for each strand separately.