ngs_week3 assignment – Xiaoyan Wen

XiaoyanWen

2/6/2022

Next Generation Sequence Analysis Homework Week 3

In this assignment, you will align short reads to the human reference genome and conduct a short exercise that uses samtools to process SAM/BAM short read alignments. This is part of multi-week "re-sequencing" workflow where the aim is to call single nucleotide polymorphisms (SNPs) from Illumina short read sequencing of human genomes.

##About the data This week we continue working with re-sequencing data from the 1000 Genomes Project. We will work with 30 adapter-trimmed and quality filtered reads (processed via fastp as in Week 2).

##Task 1: Preparing the human reference genome ###Introduction to reference genomes Reference genomes are typically FASTA-formatted genome sequences representing the primary sequence of a (typically) haploid copy of the genome. In this task, you will prepare a copy human reference genome for short read alignment with the Burrows-Wheeler Aligner BWA-MEM.

Choosing the appropriate human reference genome is complicated by the fact that many versions exist (GRCh38/hg38 is current) and for each version hundreds of variants exists. For example, there are repeat-masked, soft-masked, primary assembly and top-level assemblies (with some variable regions appearing more than once as different haplotypes, or alternate sequences) to name just a few.

we chose an appropriate variant of the hg38 version human reference genome for short read alignment from ENSEMBL. The FASTA-formatted file was then downloaded using

```
wget ftp://ftp.ensembl.org/pub/current_fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna_sm.primary_assembly.fa.gz
```

###Normalizing the sequence identifiers of a reference genome One disadvantage of FASTA format is that it does not include strict rules for sequence identifiers. Many downstream tools require that the sequence identifiers contain no white spaces (even though the FASTA specification allows whitespaces in identifiers).

####Picard-Tools NormalizeFasta program: https://broadinstitute.github.io/picard/command-line-overview.html#NormalizeFasta (https://broadinstitute.github.io/picard/command-line-overview.html#NormalizeFasta) This tool has been used to removes any white spaces in sequence identifers in the reference FASTA to avoid possible conflicts with downstream applications.

###Index files for the reference genome FASTA Many NGS software require that input files be indexed prior to including them in work flows. Index files permit rapid lookups of genome coordinates in large files and dramatically speed up computation times.

The re-sequencing workflow we will run to align reads from samples in the 1,000 genomes project followed by snp-calling requires the following index files: - a FASTA index - a set of index files required by the BWA-MEM aligner - a dictionary file required by the Genome Analysis Toolkit (GATK)

Below, you will create a FASTA index and a bwa index for the normalized reference genome. We will create a the dictionary file in a subsequence exercise.

-See here for requirements for the reference by GATK: https://gatk.broadinstitute.org/hc/en-us/articles/360035531652-FASTA-Reference-genome-format (https://gatk.broadinstitute.org/hc/en-us/articles/360035531652-FASTA-Reference-genome-format) -See samtools manual for the samtools faidx command: http://www.htslib.org/doc/samtools.html (http://www.htslib.org/doc/samtools.html) -See BWA manual for bwa index here: http://bio-bwa.sourceforge.net/bwa.shtml (http://bio-bwa.sourceforge.net/bwa.shtml)

Before beginning, log in to Greene and request a compute node for interactive use:

```
srun --time=4:00:00 --mem=4GB --pty /bin/bash
```

Create a directory in your /scratch as follows, change directories to it, and copy to it the normalized hg38 reference genome FASTA as follows:

```
cd $SCRATCH
mkdir hg38
cd hg38
cp /scratch/work/courses/BI7653/hw3.2022/hg38/Homo_sapiens.GRCh38.dna_sm.primary_assembly.normalized.fa .
```

Create a slurm job script that will create the FASTA index files in the same directory as the reference genome FASTA. You may use the template slurm script provided by your instructor in Week 1

```
/scratch/work/courses/BI7653/hw1.2022/slurm_template.sh
```

Key point: For virtually all NGS applications, the reference genome index files must appear in the same directory as the reference FASTA. Your slurm job script should do the following: - Request 32GB of memory for 5 hours. - Load the samtools and bwa modules (most recent versions on Greene) - Run samtools faidx and bwa index using the following commands on the normalized reference fasta you copied to your hg38 directory above.

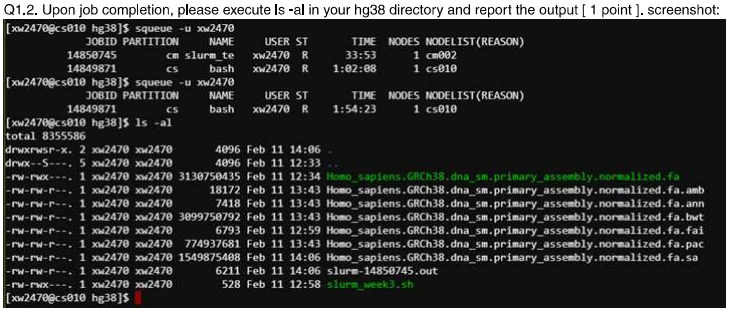
```
samtools faidx <reference fasta>
bwa index -a bwtsw <reference fasta>
```

You may now execute your job script. This should take less than 2 hours from the time the job is released from the queue. Occasionally, students have encountered "segmentation fault" errors when running this tool. If your STDERR for the job contains an error of this type, try running again.

Q1.1. Please report the contents of your job script [1 point].

```
#!/bin/bash
#
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=5:00:00
#SBATCH --mem=32GB
#SBATCH --job-name=slurm_template
#SBATCH --mail-type=FAIL
#SBATCH --mail-user=xw2470@nyu.edu

module purge

echo script begin: $(date)

module load samtools/intel/1.14
module load bwa/intel/0.7.17
samtools faidx Homo_sapiens.GRCh38.dna_sm.primary_assembly.normalized.fa
bwa index -a bwtsw Homo_sapiens.GRCh38.dna_sm.primary_assembly.normalized.fa

echo script completed: $(date)
```

Q1.2. Upon job completion, please execute ls -al in your hg38 directory and report the output [1 point]. screenshot:



Task 2: Short read alignment with BWA-MEM

BWA-MEM is a common alignment tool that was used in the 1000 Genomes Project and is the preferred tool in the Genome Analysis Toolkit SNP-calling and genotyping workflow which we will follow in this course. You may wish to read the BWA documentation http://bio-bwa.sourceforge.net/bwa.shtml (http://bio-bwa.sourceforge.net/bwa.shtml)

In this task, you will execute a job script provided by your instructor to align paired-end fastq data from 30 human samples to the human reference genome. Your instructor has prepared a set of 30 samples (60 paired-end fastqs) for you to align (paired end sequence data from the same 21 samples processed with fastp in week 2 + 9 additional paired end samples) representing four populations in the 1000 genomes project. The fastqs were processed using the same fastp command line you used in week 2 Task 2 and are ready for short read alignment with BWA-MEM.

You will now align short reads to the human reference genome using a script provided by your instructor. You are simply asked to modify one variable in the script to contain the path on your /scratch to the reference genome FASTA (see below)

Create a ngs.week3 directory and Task 2 subdirectory in your /scratch directory.

```
cd $SCRATCH
pwd
mkdir ngs.week3
cd ngs.week3
mkdir task2
cd task2
cp /scratch/work/courses/BI7653/hw3.2022/hw3_bwamem.slurm . # copy script to present workding directory
```

The script executes a job array that will find the paired-end fastqs to be aligned using a tab-delimited table with columns sample name, read 1 fastq file name, and read 2 fastq filename. You may review that file here (e.g., with less command):

```
/scratch/work/courses/BI7653/hw3.2022/fastqs.processed/hw3_fastqs.processed.txt
```

The processed fastqs are located here:

```
/scratch/work/courses/BI7653/hw3.2022/fastqs.processed
```

We will align the reads with BWA-MEM using the following settings:

```
bwa mem -M -t $SLURM_CPUS_PER_TASK -R "@RG\tID:${sample}.id\tSM:${sample}\tPL:ILLUMINA\tLB:${sample}.lb" <reference fasta> <
read1.fq> <read2.fq> > <sam>
```

Note the following: -The SLURM_CPUS_PER_TASK environmental variable will take on the value of 8 because the slurm directives at the top of the script used #SBATCH –cpus-per-task=8. -The -R argument adds an @RG read group header line to the output SAM file. This gives the sample unique identifer and information tags in the output SAM header.

Now review *hw3_bwamem.slurm script* which you copied to your */scratch* directory above. Convince yourself that #SBATCH array=1-30 directive and subsequent code will correctly process the 30 samples listed in the file

```
/scratch/work/courses/BI7653/hw3.2022/fastqs.processed/hw3_fastqs.processed.txt
```

Now modify the script to specify the ref variable. The ref variable should be set to the full path (from root = "/") to the hg38 reference genome fasta that you indexed in Task 1. If you need to, you may cd to the directory with the reference genome and use pwd to view the path to that directory from root.

Verify that you are in your Task 2 directory and execute the job script:

```
sbatch hw3_bwamem.slurm
```

Execute the following immediately after execution to check on the status of your job:

```
squeue -u xw2470
```

Q2.1 Now either take a screen shot showing your squeue command and the output (or copy the output to your homework report) [1 point]

```
[xw2470@cs011 task2]$ sbatch hw3_bwamem.slurm
Submitted batch job 14853877
[xw2470@cs011 task2]$ sequeue -u xw2470
bash: sequeue: command not found
[xw2470@cs011 task2]$ squeue -u xw2470
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
       14853877_1        cs bwamem_a   xw2470  R       0:37      1 cs029
       14853877_2        cs bwamem_a   xw2470  R       0:37      1 cs029
       14853877_3        cs bwamem_a   xw2470  R       0:37      1 cs050
       14853877_4        cs bwamem_a   xw2470  R       0:37      1 cs057
       14853877_5        cs bwamem_a   xw2470  R       0:37      1 cs059
       14853877_6        cs bwamem_a   xw2470  R       0:37      1 cs061
       14853877_7        cs bwamem_a   xw2470  R       0:37      1 cs064
       14853877_8        cs bwamem_a   xw2470  R       0:37      1 cs070
       14853877_9        cs bwamem_a   xw2470  R       0:37      1 cs072
      14853877_10        cs bwamem_a   xw2470  R       0:37      1 cs072
      14853877_11        cs bwamem_a   xw2470  R       0:37      1 cs080
      14853877_12        cs bwamem_a   xw2470  R       0:37      1 cs082
      14853877_13        cs bwamem_a   xw2470  R       0:37      1 cs082
      14853877_14        cs bwamem_a   xw2470  R       0:37      1 cs082
      14853877_15        cs bwamem_a   xw2470  R       0:37      1 cs084
      14853877_16        cs bwamem_a   xw2470  R       0:37      1 cs084
      14853877_17        cs bwamem_a   xw2470  R       0:37      1 cs096
      14853877_18        cs bwamem_a   xw2470  R       0:37      1 cs099
      14853877_19        cs bwamem_a   xw2470  R       0:37      1 cs106
      14853877_20        cs bwamem_a   xw2470  R       0:37      1 cs122
      14853877_21        cs bwamem_a   xw2470  R       0:37      1 cs125
      14853877_22        cs bwamem_a   xw2470  R       0:37      1 cs130
      14853877_23        cs bwamem_a   xw2470  R       0:37      1 cs133
      14853877_24        cs bwamem_a   xw2470  R       0:37      1 cs146
      14853877_25        cs bwamem_a   xw2470  R       0:37      1 cs146
      14853877_26        cs bwamem_a   xw2470  R       0:37      1 cs146
      14853877_27        cs bwamem_a   xw2470  R       0:37      1 cs149
      14853877_28        cs bwamem_a   xw2470  R       0:37      1 cs149
      14853877_29        cs bwamem_a   xw2470  R       0:37      1 cs157
      14853877_30        cs bwamem_a   xw2470  R       0:37      1 cs160
        14853852        cs     bash   xw2470  R       2:51      1 cs011
[xw2470@cs011 task2]$
```

This job could take 4-8 hours to complete once each of the 30 "subjobs" have executed. When your job is complete, please return to the directory where you executed the script and report the output of the following command.

```
grep _ESTATUS_ slurm-14853877*.out
```

output:

```
[xw2470@gr062 task2]$ grep _ESTATUS_ slurm-14853877*.out
slurm-14853877_10.out:_ESTATUS_ [ bwa mem for HG00149 ]: 0
slurm-14853877_11.out:_ESTATUS_ [ bwa mem for HG00260 ]: 0
slurm-14853877_12.out:_ESTATUS_ [ bwa mem for NA18907 ]: 0
slurm-14853877_13.out:_ESTATUS_ [ bwa mem for NA19137 ]: 0
slurm-14853877_14.out:_ESTATUS_ [ bwa mem for NA19093 ]: 0
slurm-14853877_15.out:_ESTATUS_ [ bwa mem for NA19256 ]: 0
slurm-14853877_16.out:_ESTATUS_ [ bwa mem for NA19098 ]: 0
slurm-14853877_17.out:_ESTATUS_ [ bwa mem for NA18870 ]: 0
slurm-14853877_18.out:_ESTATUS_ [ bwa mem for NA18909 ]: 0
slurm-14853877_19.out:_ESTATUS_ [ bwa mem for NA19138 ]: 0
slurm-14853877_1.out:_ESTATUS_ [ bwa mem for NA18757 ]: 0
slurm-14853877_20.out:_ESTATUS_ [ bwa mem for HG00151 ]: 0
slurm-14853877_21.out:_ESTATUS_ [ bwa mem for HG00106 ]: 0
slurm-14853877_22.out:_ESTATUS_ [ bwa mem for HG01914 ]: 0
slurm-14853877_23.out:_ESTATUS_ [ bwa mem for HG01985 ]: 0
slurm-14853877_24.out:_ESTATUS_ [ bwa mem for HG01986 ]: 0
slurm-14853877_25.out:_ESTATUS_ [ bwa mem for HG02013 ]: 0
slurm-14853877_26.out:_ESTATUS_ [ bwa mem for HG02051 ]: 0
slurm-14853877_27.out:_ESTATUS_ [ bwa mem for HG01879 ]: 0
slurm-14853877_28.out:_ESTATUS_ [ bwa mem for HG01880 ]: 0
slurm-14853877_29.out:_ESTATUS_ [ bwa mem for HG01896 ]: 0
slurm-14853877_2.out:_ESTATUS_ [ bwa mem for NA18627 ]: 0
slurm-14853877_30.out:_ESTATUS_ [ bwa mem for HG01915 ]: 0
slurm-14853877_3.out:_ESTATUS_ [ bwa mem for NA18591 ]: 0
slurm-14853877_4.out:_ESTATUS_ [ bwa mem for NA18566 ]: 0
slurm-14853877_5.out:_ESTATUS_ [ bwa mem for NA18644 ]: 0
slurm-14853877_6.out:_ESTATUS_ [ bwa mem for NA18545 ]: 0
slurm-14853877_7.out:_ESTATUS_ [ bwa mem for HG00113 ]: 0
slurm-14853877_8.out:_ESTATUS_ [ bwa mem for HG00243 ]: 0
slurm-14853877_9.out:_ESTATUS_ [ bwa mem for HG00132 ]: 0
```

Now recursively search the present directory and all subdirectories for files with the .sam extension:

```
find . -name \*sam   # the \* is a wild card to match anything in a file name before "sam"
```

```
[xw2470@gr062 task2]$ find . -name \*sam
./NA18591/NA18591.sam
./HG00260/HG00260.sam
./NA18909/NA18909.sam
./HG01985/HG01985.sam
./HG01879/HG01879.sam
./HG01915/HG01915.sam
./NA18627/NA18627.sam
./HG00106/HG00106.sam
./NA18644/NA18644.sam
./NA19093/NA19093.sam
./HG01880/HG01880.sam
./HG01986/HG01986.sam
./NA18907/NA18907.sam
./HG02013/HG02013.sam
./HG00149/HG00149.sam
./NA18757/NA18757.sam
./NA19098/NA19098.sam
./NA18870/NA18870.sam
./HG02051/HG02051.sam
./HG00243/HG00243.sam
./NA19137/NA19137.sam
./HG00151/HG00151.sam
./HG00132/HG00132.sam
./NA18545/NA18545.sam
./HG00113/HG00113.sam
./HG01896/HG01896.sam
./NA19256/NA19256.sam
./NA19138/NA19138.sam
./HG01914/HG01914.sam
./NA18566/NA18566.sam
[xw2470@gr062 task2]$
```

Q2.2. Please report your grep command and find commands and there outputs in your report. How many .sam files were produced? What do the exit statuses of the 30 subjobs indicate? [1 point].

answer: there are 30 .sam files were created. and the exit statuses are 0s.

##Task 3: SAM/BAM format and Samtools In this task, you will use Samtools to extract information from and manipulate a SAM/BAM file. You may wish to first watch the pre-recorded lecture on the SAM/BAM format as this provides the necessary background and review the following online documentation for Samtools and the SAM format.

-Samtools documentation: http://www.htslib.org/doc/samtools.html (http://www.htslib.org/doc/samtools.html) -SAM format specification: https://samtools.github.io/hts-specs/SAMv1.pdf (https://samtools.github.io/hts-specs/SAMv1.pdf)

Log into Greene and check out a compute node:

```
srun --time=4:00:00 --mem=4GB --pty /bin/bash
```

For questions in Task 3, you will conduct a series of operations on a BAM from the 1000 genomes project with paired-end reads mapped to chromosome 20. The path to the BAM file on Greene is:

```
/scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
```

For all answers for Task 3, please report your command lines and answer the questions. You can find assistance in the above mentioned samtools and SAM specification links and in the course lecture(s) for this week. Q3.1. Review the samtools view documentation. Then, use this program to extract only the header from the bam file above and answer the following [1 point] Q3.1a. Report your command line

```
samtools view -H /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
```

Q3.1b. Report the @HD header tag line. What does the information in this line indicate?

```
@HD     VN:1.0  SO:coordinate
```

Answer: it means this bam file is sorted.

Q3.2. Use samtools view to answer the following. Review samtools view options -c, -f, and -F. Please answer the following questions including (1) your command line you used to obtain the answer and (2) the output written to your terminal [1 point].

Q3.2a how many unmapped reads are there in the BAM (hint: use appropriate bitwise flag(s) described in SAM/BAM lecture and documentation?

```
samtools view -c -f 4 /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
## output: 7247
```

Q3.2b How many mapped reads are there in the BAM?

```
samtools view -c -F 4 /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
## output: 2924253
```

Q3.2c What is the percentage mapping rate (total mapped reads / total reads in the alignment) for this sample?

```
## 99.75% mapping rate
```

Q3.3. A hypothetical SAM file has alignment records with the bitwise flag values that include 4, 147, 113, 99 on the decimal scale. What are the binary and hexadecimal representations of each of the these values? [1 point]. Hint: https://www.rapidtables.com/convert/number/binary-to-hex.html (https://www.rapidtables.com/convert/number/binary-to-hex.html)

```
Answer:
##          Binary        Hexadecimal
## 4 --     100           4
## 147 --   10010011      93
## 113 --   1110001       71
## 99 --    1100011       63
```

Q3.4. Picard has an online tool for determining the meanings of bitwise flag values such as those in Q3.3: http://broadinstitute.github.io/picard/explain-flags.html (http://broadinstitute.github.io/picard/explain-flags.html) Using this tool, what are the characteristics of the reads in each these four flags in Q3.3 (4,147,113,99)? [1 point].

```
Answer:
# # 4 - read unmapped
# # 147 - read paired, read mapped in proper pair, read reverse strand, scond in pair
# # 113 - read paired, read reverse strand, mate reverse strand, first in pair
# # 99 - read paired, read mapped in proper pair, mate reverse strand, first in pair
```

Q3.5. SAM specification allows for three types of alignment records. There are primary alignments, secondary alignments and supplementary alignments. Depending on the alignment software and command line used, secondary or supplementary alignments may also exist in a SAM/BAM file.BWA-MEM will typically add both primary and secondary alignments. Note that by using the -M option above we instructed bwa to set all supplementary alignments to secondary (0x100).

For Q3.5, use samtools view with appropriate -c, -f, -F options to count the following in the BAM file at

```
/scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
```

For each answer, provide the number of reads and the command line you used.Now answer the following [1 point]: Q3.5a How many alignments are primary?

```
samtools view -c -F 256 /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
## output: 2931500
```

Q3.5b How many alignments are secondary?

```
samtools view -c -f 256 /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
## output: 0
```

Q3.5c How many alignments are supplementary

```
samtools view -c -f 2046 /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
## output: 0
```

Q3.5d What is the number of reads excluding unmapped reads, supplementary reads, secondary reads and PCR duplicates?

```
samtools view -c -F 3332 /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
## output: 2885340
```

Q3.6. A common task is to subset a SAM/BAM to include a subset of positions on a chromosome. Use *samtools view* to subset the BAM from Q3.5 from chromosome 20 position 1 to 2000000 (i.e., 2 Mb), while also retaining the header. Note that to perform this type of operation, the BAM must be coordinate-sorted, (which it is). Hint: See http://www.htslib.org/doc/samtools.html (http://www.htslib.org/doc/samtools.html)

Now count the reads in the subsetted BAM. Report the command line used to subset the BAM and the number of reads in the subset [1 point].

```
samtools view -c /scratch/work/courses/BI7653/hw3.2022/HG00096.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam 20:1-20000
00
## output: 95338
```