SAMtools mpileup / VCF/BWA/BCF VARIANT CALLING

Folder: gencommand\_proj3\_data

Module 3 Exam – Command Line Tools for Genomic DS

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Step 1: Suppose the current directory is ‘/media/sf\_gencommand\_proj3\_data/’, which is where supporting data for Exam 3 are stored on the virtual machine for this course. First, we generate the bowtie2 index for the genome file provided. We create a sub-directory ‘wu\_0’ to store the index, then invoke ‘bowtie2-build’:

% mkdir wu\_0

% bowtie2-build wu\_0.v7.fas /media/sf\_gencommand\_proj3\_data/wu\_0/wu\_0

Notes: If your work directory is different, substitute it in the command above. Also, the 'bt2\_index\_base' in the bowtie2-build command line usage should consist of both the path to the index directory and the prefix used for the index files.As a reminder, you can use:

% bowtie2-build >& bowtie2-build.log

to save and then review the command line usage for the program and decide on the relevant parameters.Step 2: To answer the question, then simply inspect the content of the index directory and directly observe the number of index files

% ls wu\_0/

A screenshot of a computer

Description automatically generated

List the files in the index directory and observe their extension (n.b., all index files have the extension ‘bt2’):

% ls wu\_0/

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

Step 1: We first run bowtie2 with two sets of parameters: i) the default parameters, to generate end-to-end read alignments; and ii) the ‘--local’ option, to produce potential partial alignments of a read. For both runs, display the output as SAM alignments (option ‘-S’):

% bowtie2 –x wu\_0/wu\_0 –U wu\_0\_A\_wgs.fastq –S out.full.sam

% bowtie2 –x wu\_0/wu\_0 –U wu\_0\_A\_wgs.fastq –S out.local.sam --local

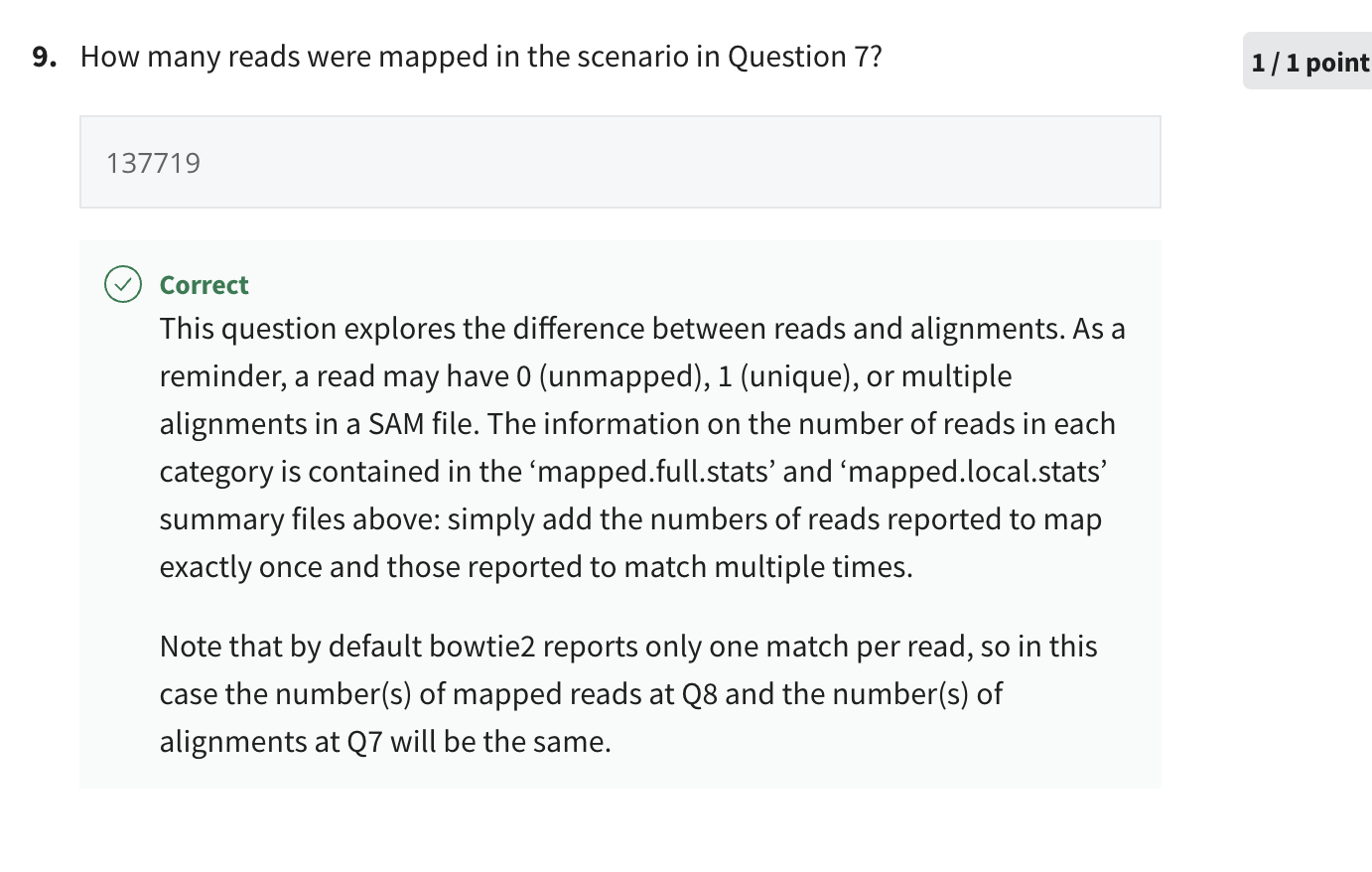
These will create the SAM files ‘out.full.sam’ and ‘out.local.sam’. Upon completion, each run also prints a set of summary statistics on the number of reads unmapped/mapped exactly once/mapped multiple times. Save these to local files, say ‘mapped.full.stats’ and ‘mapped.local.stats’.Step 2: To answer question Q7, we inspect the SAM files created and determine the number of alignment lines, excluding lines that refer to unmapped reads. A SAM line indicating an unmapped read can be recognized by a ‘\*’ in column 3 (chrom). Additionally, we need to exclude the SAM header:

% cat out.full.sam | grep –v “^@” | cut –f3 | grep –v “\*” | wc –l

% cat out.local.sam | grep –v “^@” | cut –f3 | grep –v “\*” | wc –l

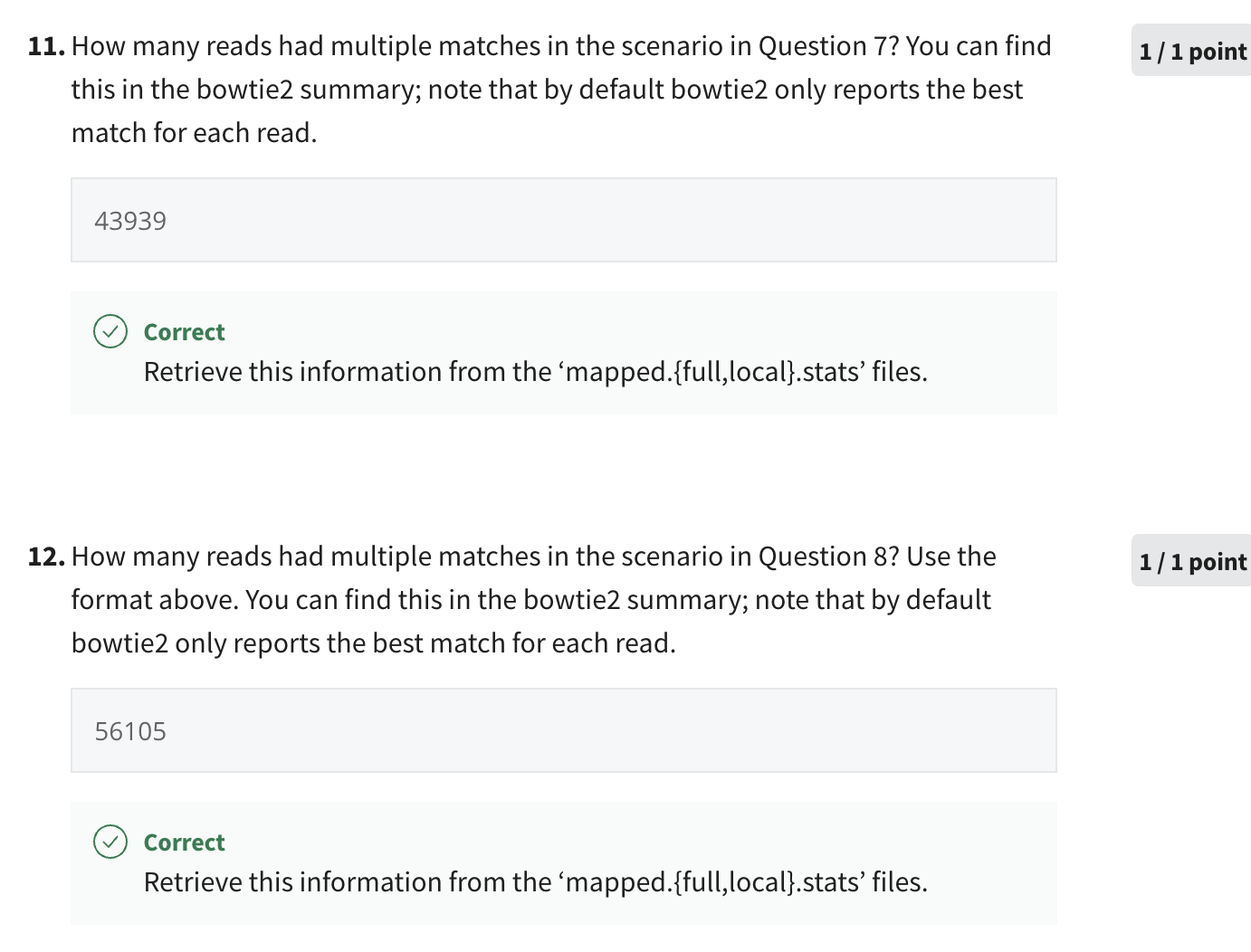
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



% bowtie2 -p 4 –x wu\_0/wu\_0 wu\_0\_A\_wgs.fastq –S wu\_0.bt2.sam

A screenshot of a computer

Description automatically generated

This information is captured in the CIGAR field, marked with ‘D’ and ‘I’, respectively:

% cut –f6 out.full.sam | grep –c “[I,D]”

% cut –f6 out.local.sam | grep –c “[I,D]”

A screenshot of a computer

Description automatically generated

This information is captured in the CIGAR field, marked with ‘D’ and ‘I’, respectively:

% cut –f6 out.full.sam | grep –c “[I,D]”

% cut –f6 out.local.sam | grep –c “[I,D]”

A screenshot of a computer

Description automatically generated

Step 1: Start by converting the SAM file to BAM format as indicated:

% samtools view –bT wu\_0.v7.fas out.full.sam > out.full.bam

then sorting it:

% samtools sort out.full.bam out.full.sorted

This will create the BAM file ‘out.full.sorted.bam’, which will be used to determine sites of variation.Step 2: Determine candidate sites using ‘samtools mpileup’, providing the reference fasta genome (option ‘-f’) and using the option ‘-uv’ to report the output in uncompressed VCF format:

% samtools mpileup –f wu\_0.v7.fas –uv out.full.sorted.bam > out.full.mpileup.vcf

Step 3: Count the number of entries in the VCF file located on Chr3. The chromosome information is listed in column 1, once we filter out the header lines (marked with “#”):

1

% cat out.full.mpileup.vcf | grep –v “^#” | cut –f1 | grep –c “^Chr3”

Potential pitfalls: It is critical to sort the BAM file before analyzing it with samtools.

A screenshot of a computer

Description automatically generated

This information is contained in column 4:

1

% cat out.full.mpileup.vcf | grep –v “^#” | cut –f4 | grep –P “^A$”

where “^A$” tells ‘grep’ to look for patterns that consist exclusively of ‘A’ (i.e., between the start ‘^’ and end ‘$’ of the line).

A close-up of a text

Description automatically generated

Read depth is indicated by the ‘DP=’ field in column 8:

1

% cat out.full.mpileup.vcf | grep –v “^#” | grep –c “DP=20;”

A white rectangular object with a white background

Description automatically generated

This information is marked with the keyword ‘INDEL’ in the variant line:

1

% cat out.full.mpileup.vcf | grep –v “^#” | grep –c INDEL

A screenshot of a phone

Description automatically generated

This information is stored in columns 1 and 2 of the VCF file:

1

% cat out.full.mpileup.vcf | grep –v “^#” | cut –f1,2 | grep Chr1 | grep 175672

then select only the entries corresponding to position 175672.

A question mark on a white background

Description automatically generated

Step 1: First re-run ‘SAMtools mpileup’ with the BCF output option ‘-g’:

1

% samtools mpileup –f wu\_0.v7.fas –g out.full.sorted.bam > out.full.mpileup.bcf

then call variants using ‘BCFtools call’ with the multi-allelic caller (option ‘-m’), showing only variant sites (‘-v’) and presenting the output in uncompressed VCF format (‘-O v’), as instructed:

1

% bcftools call –m –v –O v out.full.mpileup.bcf > out.final.vcf

Step 2: To answer the question, we count all reported variants that show ‘Chr3’ in column 1:

1

% cat out.final.vcf | grep –v “^#” |  cut –f1 | sort | uniq –c | grep “Chr3”

A screenshot of a phone

Description automatically generated

This information is stored across columns 2 and 3. An A->T SNP would be represented as an ‘A’ in column 2 and a ‘T’ in column 3:

1

% cat out.final.vcf | grep –v “^#” | cut –f4,5 | grep –P “^A\tT$” | wc -l

A white rectangular object with a white background

Description automatically generated

Similar to Q14, indels are marked with ‘INDEL’:

1

% cat out.final.vcf | grep –v “^ #” | grep –c INDEL

A screenshot of a phone

Description automatically generated

Similar to Q13:

1

% cat out.final.vcf | grep –v “^#” | grep –c “DP=20;”

A screenshot of a phone

Description automatically generated

Similar to Q15:

1

% cat out.final.vcf | grep –v “^#” | cut -f1-5 | grep Chr3 | grep 11937923

then inspect the reference and variant sequences in columns 4 and 5 to determine the type of variant.