MAKERERE UNIVERSITY

COLLEGE OF HEALTH SCIENCES

MSc BIOINFORMATICS

NABUKEERA LYDIA

2022/HD05/1969

BASH ASSIGNMENT

You are provided with two files (A VCF and a SAM file). The files can be downloaded from here (https://drive.google.com/drive/folders/11UD52i99CaCSBEJFNb8Y1afo9p3hL8cL?usp=sharing). Use BCFtools, SAMtools and bash utilities to answer the questions that follow.

Make your submissions on your GitHub and send the link to the repo via email to ibra.lujumba@gmail.com. Your submissions should include commands/scripts used to obtain answers to the questions as well as answers to the questions.

Deadline: 26th January 2023

Manipulating VCF files

1. **Describe the format of the file and the data stored**

A VCF (Variant Call Format) file is a text-based format used to store genetic variation data, such as single nucleotide polymorphisms (SNPs), insertions and deletions (indels), and structural variants. The format consists of a header section that contains meta-information about the file and the data it contains, followed by a column header line, and then one or more lines of data for each variant. The header section of a VCF file contains lines starting with "##" and includes information such as the file format version, the reference genome used, and the name of the software that generated the data.

2. **What does the header section of the file contain**

The header section of a VCF (Variant Call Format) file contains meta information about the file and the data it contains. It starts with lines that begin with "##" and includes key-value pairs separated by "=". The information in the header section can be grouped into several categories such as File format and version, Reference genome, and software and pipeline used to generate the data. It includes information about the version of the VCF format used in the file, the reference genome that was used to generate the variants in the file, the assembly version and the source of the genome, and the name of the software and pipeline used to generate the data.

3. **How many samples are in the file**

There were 6 samples.
Code used: bcftools query -l sample.vcf.gz | wc -l

4. **How many variants are in the file**

Using the code “bcftools query -f '%ALT\n' sample.vcf | wc –l”,
There are 398246 variants.

5. **How would you extract the chromosome, position, QualByDepth and RMSMappingQuality fields? Save the output to a tab-delimited file**

Code used: bcftools query -f '%CHROM\t%POS\t%INFO/QD\t%INFO/MQ\n' sample.vcf > QR.txt
The output is in the file named “QR.txt”.

6. **Extract data that belongs to chromosomes 2,4 and MT**

Code used: awk '$1 ~ /^(2|4|MT)$/ {print $0}' sample.vcf > chrom24MT.vcf
The output was in the output file chrom24MT.vcf

7. **Print out variants that do not belong to chr20:1-30000000**

Code used: awk '$1 != "20" || ($1 == "chr20" && ($2 < 1 || $2 > 30000000)) \
{> {print $1, $2, $4, $5}' sample.vcf > Chr20.vcf

The output is in file Chr20.vcf

8. **Extract variants that belong to SRR13107019**

Code used: bcftools query -f '%CHROM\t%POS\t%REF\t%ALT\n' -s SRR13107019 sample.vcf > SRR19variant.txt

The output is named “SRR19variant.txt”

9. **Filter out variants with a QualByDepth above 7**

Code used: vcftools --vcf sample.vcf --minDP 7 --recode --out QD7

And the output file was specified to be “QD7”

After filtering, kept 398246 out of a possible 398246 Sites

10. **How many contigs are referred to in the file. Check the header section**

Code used: grep -c "^##contig" sample.vcf

The anwer is 2211 contigs.

11. **Comment on the eighth and ninth columns of the file**

The eighth column of a VCF file typically contains information about the genotype of the sample being called, while the ninth column and following columns contain the sample-specific information, including the genotype likelihoods, the phred-scaled quality of the genotype call, the depth of coverage and other annotation tags. The genotype is represented in format of "GT:AD:DP:GQ:PL" where GT is the genotype call, AD is the read depth for each allele, DP is the read depth for this genotype, GQ is the genotype quality and PL is the phred-scaled genotype likelihoods.

12.

13.

14. **Extract data on the read depth of called variants for sample SRR13107018**

Code used: bcftools query -f '%DP\n' -s SRR13107018 sample.vcf > readSRR18

The output file is readSRR18

15. **Extract data on the allele frequency of alternate alleles. Combine this data with the chromosome and position of the alternate allele**

Code used: bcftools query -f '%CHROM\t%POS\t%AF\n' sample.vcf > ALTallele.txt

The output file name is “ALTallele.txt”

Manipulating SAM files

1. **Describe the format of the file and the data stored**

The sequence Alignment/Map (SAM) is a text-based file format that stores the alignment of single- and paired-end reads produced by different sequencing platforms. All lines in a SAM file are tab-delimited, and mandatory fields must be present with their value either being a * or an 0 depending on the field.

2. **What does the header section of the file contain?**

The header section contains information about the entire file and additional information for alignments. This section begins with the @ symbol, distinguishing them from the alignment section.

3. **How many samples are in the file**

Code used: grep -E '^@RG' sample.sam | awk '{print $2}' | awk -F ':' '{print $2}' | sort | uniq | wc –l

They are 249 samples.

4. **How many alignments are in the file**

Code used: samtools view -F 4 sample.sam | wc –l

There are 35511 alignments.

5. **Get summary statistics for the alignments in the file**

Code used: samtools flagstat sample.sam > samstat.txt

Output file is named “samstat.txt”

6. **Count the number of fields in the file**

Code used: awk '{print NF}' sample.sam

There are 18 fields

7. **Print all lines in the file that have @SQ and sequence name tag beginning with NT_**

Code used: grep '@SQ.*NT_' sample.sam

8. **Print all lines in the file that have @RG and LB tag beginning with Solexa**

Code used: grep "@RG.*LB:Solexa" sample.sam

9. **Extract primarily aligned sequences and save them in another file**

Code used: awk '$1 !~ /^@/ && $2 == "99" || $2 == "83"' sample.sam > savedalignedseq.sam

10. **Extract alignments that map to chromosomes 1 and 3. Save the output in BAM format**

awk '$1 !~ /^@/ && ($3 == "1" || $3 == "3")' sample.sam | samtools view -Sb - > alignmap.bam

11. **How would you obtain unmapped reads from the file**

Code used: samtools view -f 4 sample.sam > unmappedreads.sam

Output file: “unmappedreads.sam”

12. **How many reads are aligned to chromosome 4**

Code used: grep -c "^4\t" sample.sam

There were no aligned reads.

13. **Comment on the second and sixth columns of the file**

The second column of a SAM file typically contains the name of the reference sequence and in the given file it is the read aligns to. The sixth column contains the CIGAR string, which describes the alignment of the read to the reference sequence.

14. **Extract all optional fields of the file and save them in "*optional_fields.txt*"**

Code used: awk '{for(i=11;i<=NF;i++) print $i}' sample.sam > optional_fields.txt