TECHNICAL REPORT ON VARIANT CALLING ANALYSIS

The most important question surrounding this task is “how do I find sequence variants between a given sample and a reference genome?” To answer this question, the tasks describe the steps involved in variant calling, the types of data formats encountered during variant calling and the command line tools used to perform variant calling are described in this report.

Two samples were used for this analysis, with each of the samples containing the forward and reverse strand.

Methodology

The methodology involved in variant calling will follow the workflow as shown in figure 1.

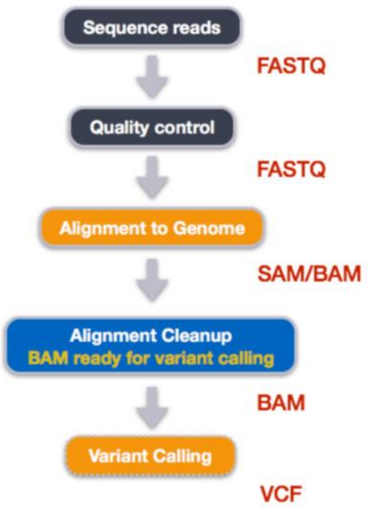


Figure 1: Variant calling workflow

Setting up

From the home directory, a new directory called stageII was created and changed into that directory using the following command.

```
mkdir stageII && cd stageII
```

Software installation

All the software involved in this analysis were installed on the terminal using the following commands respectively.

- sudo apt-get install fastp for trimming out adapters from reads
- sudo apt-get install fastqc for carrying out quality checks on the reads
- sudo apt-get install bwa for aligning the samples to a reference genome
- sudo apt-get install samtools for visualizing the alignment files
- conda install -c bioconda -c conda-forge multiqc for getting a comprehensive report of the quality checked reads
- sudo apt-get install bcftools for detecting Single Nucleotide Variations (SNV's)

Creation of Directories

Several directories and folders were created for ease of the variant calling workflow.

- A new directory called daatah was created, within which, another folder called reference_G was created to house the reference genome using the following command;

  ```
  mkdir -p daatah/reference_G
  ```

- Another directory called qc_reads was created using `mkdir qc_reads` so as to house the results of the quality checked samples.
- Since alignment usually generates sam, bam, bcf and vcf files, some folders were created to contain these files using the following command;

  ```
  mkdir -p resolts/sam resolts/bam resolts/bcf resolts/veecf
  ```

Pre-alignment analysis

To download the reference genome into the already created daatah/reference_G directory, the following command line was used

```
wget https://zenodo.org/record/2582555/files/hg19.chr5_12_17.fa.gz -o daatah/reference_G/hg19.chr5_12_17.fa.gz
```

However, the it was observed that the reference genome was in a zipped format. Therefore, to gunzip the reference genome, the following command was used;

```
gunzip daatah/reference_G/hg19.chr5_12_17.fa.gz
```

After the reference genome has been downloaded and gun zipped, the sample data files to be analysed were downloaded using the following commands respectively for each dataset

```
wget https://zenodo.org/record/2582555/files/SLGFSK-N_231335_r1_chr5_12_17.fastq.gz -O SLGFSK-N_231335_r1_chr5_12_17.fastq.gz
wget https://zenodo.org/record/2582555/files/SLGFSK-N_231335_r2_chr5_12_17.fastq.gz -O SLGFSK-N_231335_r2_chr5_12_17.fastq.gz
wget https://zenodo.org/record/2582555/files/SLGFSK-T_231336_r1_chr5_12_17.fastq.gz -O SLGFSK-T_231336_r1_chr5_12_17.fastq.gz
wget https://zenodo.org/record/2582555/files/SLGFSK-T_231336_r2_chr5_12_17.fastq.gz -O SLGFSK-T_231336_r2_chr5_12_17.fastq.gz
```

To ensure that the adapters on each of the sample data sets have been trimmed, fastp was used for trimming based on the following command for each of the samples;

```
fastp -i SLGFSK-N_231335_r1_chr5_12_17.fastq.gz -I SLGFSK-N_231335_r2_chr5_12_17.fastq.gz -o SLGFSK-N_231335_r1_chr5_12_17.fastq.gz -O SLGFSK-N_231335_r2_chr5_12_17.fastq.gz

fastp -i SLGFSK-T_231336_r1_chr5_12_17.fastq.gz -I SLGFSK-T_231336_r2_chr5_12_17.fastq.gz -o SLGFSK-T_231336_r1_chr5_12_17.fastq.gz -O SLGFSK-T_231336_r2_chr5_12_17.fastq.gz
```

To check the quality of the reads, in order to ensure that they all passed a certain threshold, fastqc was used based on the following command and outputs into the qc_reads folder

```
fastqc SLGFSK-N_231335_r1_chr5_12_17.fastq.gz -o qc_reads/
fastqc SLGFSK-N_231335_r2_chr5_12_17.fastq.gz -o qc_reads/
fastqc SLGFSK-T_231336_r1_chr5_12_17.fastq.gz -o qc_reads/
fastqc SLGFSK-T_231336_r2_chr5_12_17.fastq.gz -o qc_reads/
```

Alignment

Prior to carrying out burrow wheeler alignment, the reference genome was indexed using

```
bwa index daatah/reference_G/hg19.chr5_12_17.fa
```

After indexing the reference genome, the samples were aligned with the reference genome using the following command;

```
bwa mem daatah/reference_G/hg19.chr5_12_17.fa SLGFSK-N_231335_r1_chr5_12_17.fastq SLGFSK-N_231335_r2_chr5_12_17.fastq.gz > resolts/sam/SLGFSK-N.aligned.sam
bwa mem daatah/reference_G/hg19.chr5_12_17.fa SLGFSK-T_231336_r1_chr5_12_17.fastq SLGFSK-T_231336_r2_chr5_12_17.fastq > resolts/sam/SLGFSK-T.aligned.sam
```

Post alignment analysis

Since the sam files are heavy files, there's need to convert them into Binary Alignment Maps using the following commands

```
samtools view -S -b resolts/sam/SLGFSK-N.aligned.sam > resolts/bam/SLGFSK-N.aligned.bam
samtools view -S -b resolts/sam/SLGFSK-T.aligned.sam > resolts/bam/SLGFSK-T.aligned.bam
```

To Sort BAM files by coordinates, the following commands were used for each of the samples

```
samtools sort -o resolts/bam/SLGFSK-N.aligned.sorted.bam resolts/bam/SLGFSK-N.aligned.bam
samtools sort -o resolts/bam/SLGFSK-T.aligned.sorted.bam resolts/bam/SLGFSK-T.aligned.bam
samtools flagstat resolts/bam/SLGFSK-N.aligned.sorted.bam
samtools flagstat resolts/bam/SLGFSK-T.aligned.sorted.bam
```

Prior to detecting the SNV's, the read coverage of positions in the genome were calculated using

```
bcftools mpileup -O b -o resolts/bcf/SLGFSK-N_raw.bcf --no-reference -f hg19.chr5_12_17.fa resolts/bam/SLGFSK-N.aligned.sorted.bam
bcftools mpileup -O b -o resolts/bcf/SLGFSK-T_raw.bcf --no-reference -f hg19.chr5_12_17.fa resolts/bam/SLGFSK-T.aligned.sorted.bam
```

Finally, to detect the single nucleotide variants (SNVs) for each sample, the following command was used,

```
bcftools call --ploidy 1 -m -v -o resolts/veecf/SLGFSK-N_variants.veecf resolts/bcf/SLGFSK-N_raw.bcf
bcftools call --ploidy 1 -m -v -o resolts/veecf/SLGFSK-T_variants.veecf resolts/bcf/SLGFSK-T_raw.bcf
```

In order to filter and report the SNV variants in variant calling format (VCF), the following was used;

```
vcfutils.pl varFilter resolts/veecf/SLGFSK-N_variants.veecf > resolts/veecf/SLGFSK-N_final_variants.veecf
```

Using the following commands, the vcf formats were explored

```
less -S resolts/veecf/SLGFSK-N_final_variants.veecf
less -S resolts/veecf/SLGFSK-T_final_variants.veecf
```

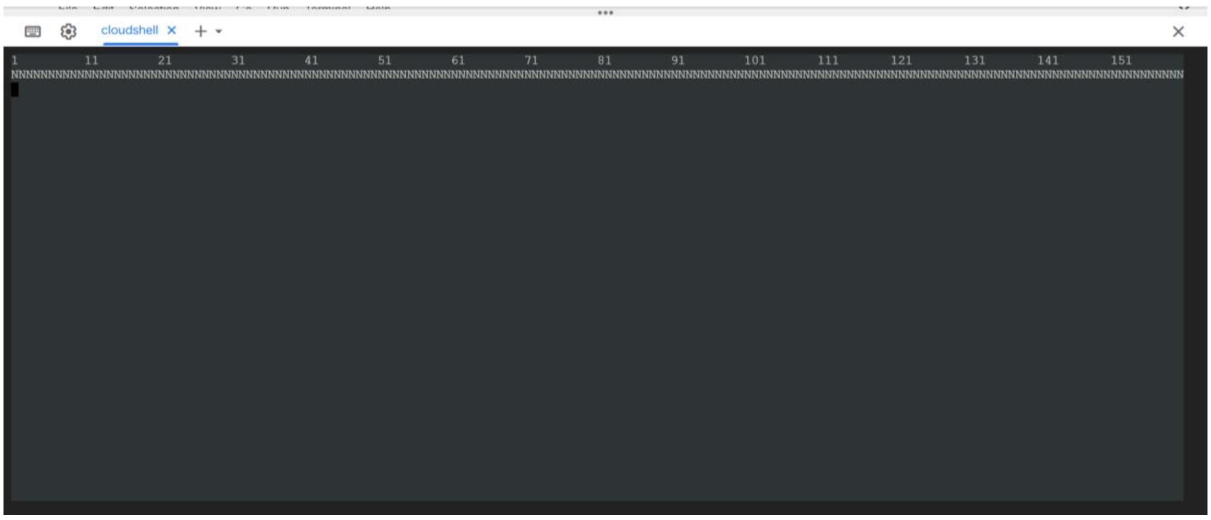
The next step was to visualise the alignment files. Before then, the BAM file was indexed using the following command

```
samtools index resolts/bam/SLGFSK-N.aligned.sorted.bam
samtools index resolts/bam/SLGFSK-T.aligned.sorted.bam
```

Using tview, the aligned and sorted bam files were viewed using;

```
samtools tview resolts/bam/SLGFSK-N.aligned.sorted.bam
samtools tview resolts/bam/SLGFSK-T.aligned.sorted.bam
```

Results



```
1         11        21        31        41        51        61        71        81        91        101       111       121       131       141       151
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

Figure 2: Showing the output of tview, highlighting the mapped reads against a reference genome

Conclusion

This report highlights the steps involved in variant calling, the types of data formats encountered during variant calling and the command line tools used to perform variant calling. The sample datasets used are atypical, that explains the non-clarity of results shown in figure 2.

The script to this pipeline can be found here

Genomics-Ishaku/CompletedSTAGE2.sh at main · ishakulemu/Genomics-Ishaku (github.com)

Compiled by Ishaku