SOMATIC VARIANT IDENTIFICATION FROM TUMOR AND NORMAL SAMPLE PAIRS

About

This task was assigned for the stage two of the 2022 July/August HackBio Genomics cohort

Name of Participant:

Annabel Sharon Anyang

Overview

The aim of the stage 2 task was to build on the participants understanding and usage of some of the core coding and bioinformatics skills learnt in the previous weeks through practical genomics analysis.

The task is designed to introduce the tools, data types and workflow of variant identification.

Task Objectives

- Understand the steps in variant calling
- Describe the types of data formats encountered during variant calling
- Use command line to perform variant calling

Skillmap

BASH, Linux, Command Line/Terminal Usage, Git and GitHub, Molecular Biology, NGS data analysis

Background

A variant call is a report of a difference in nucleotide between a sample vs. a reference at a given position in an individual genome or transcriptome (Lawrence, M., 2014). Variant calling is the identification of single nucleotide polymorphisms (SNPs) and small insertions/deletions (Indels) from next generation sequencing (NGS) data. The identification of genomic variants can play an important role in scientific discovery (Khalfan, M., 2022).

Analysis Questions

How do I find sequence variants between my samples and a reference genome?

Methodology

Data:

The dataset used in this task was gotten from Zenodo (Maier, W., 2019). They include the forward and reverse reads sequence data from a patient's normal tissue, and from the same patient's tumor tissue **(Table 1)**. The reference genome used is the **hg19** version of the sequences of human chromosomes 5, 12 and 17.

Table 1. Dataset table of normal and tumor samples

Sample No.	**Samples**	**Type**
1	SLGFSK-N_231335_r1_chr5_12_17	**Normal**
2	SLGFSK-N_231335_r2_chr5_12_17	**Normal**
3	SLGFSK-T_231335_r1_chr5_12_17	**Tumor**
4	SLGFSK-T_231335_r2_chr5_12_17	**Tumor**

Software Packages:

Fastp, Fastqc, Multiqc, samtools, bwa, bcftools

Analysis

Step 1: Download the dataset to work with

First, for dataset download, wget command was used to retrieve the data from Zenodo in a terminal in a command-line as in:

```
wget https://zenodo.org/record/2582555/files/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
wget https://zenodo.org/record/2582555/files/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
```

Then the reference genome downloaded and gunzipped it.

```
wget https://zenodo.org/record/2582555/files/hg19.chr5_12_17.fa.gz
```

Step 2: Pre-Processing of dataset

FastQC was implemented to run a quality control on all the fastq data sample files. MultiQC was then implemented to assemble the outputted qc reports from the FastQc run of the previous step. The reads were then trimmed using a fastp bash script.* Next, multiQC was run on the resulting trimmed reads.

Step 3: Index reference genome for use by Burrow-Wheeler Aligner (BWA)

Indexing allows the aligner to quickly find the potential alignment sites for query sequences in a genome, which saves time during alignment. Indexing the reference is a one time step. The only reason to create a new index is if there is analysis involving a different reference genome or a different alignment tool. An index of reference file was created using the bwa code below as shown below:

```
bwa index hg19.chr5_12_17.fa
```

The next thing to do is to align the reads to the reference genome. The alignment process consists of choosing an appropriate reference genome to map reads against and then picking an aligner. Here, BWA-MEM to implement the alignment. Bwa was downloaded using bioconda install. The codes are shown below:

```
bwa mem -R '@RG\tID:231335\tSM:Normal' reference/hg19.chr5_12_17.fa trimmed_reads/SLGFSK-N_231335_r1_chr5_12_17.fastq.gz trimmed_reads/SLGFSK-N_231335_r2_chr5_12_17.fastq.gz > map_output/SLGFSK-N_231335.sam
```

```
bwa mem -R '@RG\tID:231336\tSM:Tumor' reference/hg19.chr5_12_17.fa trimmed_reads/SLGFSK-T_231336_r1_chr5_12_17.fastq.gz trimmed_reads/SLGFSK-T_231336_r2_chr5_12_17.fastq.gz > map_output/sam/SLGFSK-T_231336.aligned.sam
```

The SAM file is a tab-delimited text file containing individual read and genome alignment information. The compressed binary version of the SAM file is the BAM file. To reduce size and allow for indexing (this enables efficient random access of the data contained in the file), we converted the resulting output SAM files to bam files using samtools (see codes below).

```
samtools view -S -b map_output/sam/SLGFSK-N_231335.aligned.sam > map_output/bam/SLGFSK-N_231335.aligned.bam
```

```
samtools view -S -b map_output/sam/SLGFSK-T_231336.aligned.sam > map_output/bam/SLGFSK-T_231336.aligned.bam
```

The bam files were then sorted by coordinates. To view the statistics of the sorted bam files, flagstat was used (see codes below).

```
samtools sort -o map_output/bam/SLGFSK-N_231335.aligned.sorted.bam map_output/bam/SLGFSK-N_231335.aligned.bam
```

```
samtools sort -o map_output/bam/SLGFSK-N_231335.aligned.sorted.bam map_output/bam/SLGFSK-N_231335.aligned.bam
```

```
samtools flagstat map_output/bam/SLGFSK-N_231335.aligned.sorted.bam
```

```
samtools flagstat map_output/bam/SLGFSK-T_231336.aligned.sorted.bam
```

Step 4: variant Calling (VCF) workflow

For variant calling in this workflow, we used bcftools. There are many other tools available for variant calling. There were several steps carried before actually calling the variants.

I. Calculate the read coverage of positions in the genome

The bcftools package was installed using bioconda. Then the first pass on variant calling by counting read coverage with bcftools using the **mpileup** command. This step generated a file with coverage information for every base.

```
bcftools mpileup -O b -o results/bcf/ SLGFSK-N_231335_raw.bcf -f ../reference/hg19.chr5_12_17.fa map_output/bam/SLGFSK-T_231335.aligned.sorted.bam
```

```
bcftools mpileup -O b -o results/bcf/ SLGFSK-N_231336_raw.bcf -f ../reference/hg19.chr5_12_17.fa map_output/bam/SLGFSK-T_231336.aligned.sorted.bam
```

II. Detect the single nucleotide variants (SNVs)

To identify the SNVs, we used the bcftools **call** command.

```
bcftools call --ploidy 1 -m -v -o results/vcf/SLGFSK-N_231335_variants.vcf results/bcf/SLGFSK-N_231335_raw.bcf
```

```
bcftools call --ploidy 1 -m -v -o results/vcf/SLGFSK-N_231336_variants.vcf results/bcf/SLGFSK-N_231336_raw.bcf
```

III. Filter and report the SNV variants in variant calling format (VCF)

```
vcfutils.pl varFilter results/vcf/SLGFSK-N_231335_variants.vcf > results/vcf/SLGFSK-N_2331335_final_variants.vcf
```

```
vcfutils.pl varFilter results/vcf/SLGFSK-N_231336_variants.vcf > results/vcf/SLGFSK-N_2331336_final_variants.vcf
```

To explore the resulting VCF format, the **less** command was used.

```
less -S results/SLGFSK-N_231335_final_variants.vcf
```

```
less -S results/SLGFSK-N_231336_final_variants.vcf
```

Then to assess how many variants are in the vcf file, **grep** and **wc -l** commands was used to output the number of variants in the file.

```
grep -v "#" results/vcf/SLGFSK-N_2331335_final_variants.vcf | wc -l
```

To visualize the alignment, the BAM file was first indexed with samtools. In order to visualize the mapped reads, we used **tview**, a text alignment viewer. (see codes below)

```
samtools index results/bam/SLGFSK-N_2331335.aligned.sorted.bam
```

```
samtools index results/bam/SLGFSK-N_2331336.aligned.sorted.bam
```

The more elaborate codes of this pipeline can be found at this GitHub repository.

Results

The variant calling pipeline ran successfully

Conclusion

With a simple variant calling workflow, the excellent nature of this analysis in the detection of somatic variants has been highlighted. Variant calling may also be applied to distinguish somatic and germline mutations.

References

Lawrence, M (2014). Retrieved 14 August 2022, from https://bioconductor.org/help/coursematerials/2014/CSAMA2014/3_Wednesday/lectures/VariantCallingLecture.pdf

Khalfan, M (2022). Retrieved 14 August 2022, from https://learn.gencore.bio.nyu.edu/variant-calling/

Maier, W, (2019). Training data for \’Somatic variant calling\’ tutorial _Galaxy training material) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.2582555

Tutorial reference: Variant Calling Workflow – Data Wrangling and Processing for Genomics. (2022). Retrieved 14 August 2022, from https://datacarpentry.org/wrangling-genomics/04-variant_calling/index.html#overview