<u>**STAGE TWO TASK SUMMARY REPORT**</u>

TEAM-DARWIN: *@ankitamurmu009(Team Lead), @ayanotemitope, @aswanirenu, @alaberehafsat, @ anthonyboluwatife08*

IDENTIFICATION OF SOMATIC AND GERMLINE VARIANTS FROM TUMOR AND NORMAL SAMPLE PAIRS

1.0 INTRODUCTION

A mutation is a change that occurs in our DNA sequence due to genetic or environmental factors. Cancer is one of the most common diseases which occurs due to mutations in genes that control the way our cells grow and divide. These mutations can be of two types: germline mutations (inherited from either parent) and somatic mutations (which are not inherited). In addition, a common genetic mutation occurring as a loss of one normal copy or a group of genes known as Loss of Heterozygosity (LOH) is common in cancer development as well. Therefore, identifying variants is important because it can help in the early detection and prevention of cancer and also determine effective therapies.

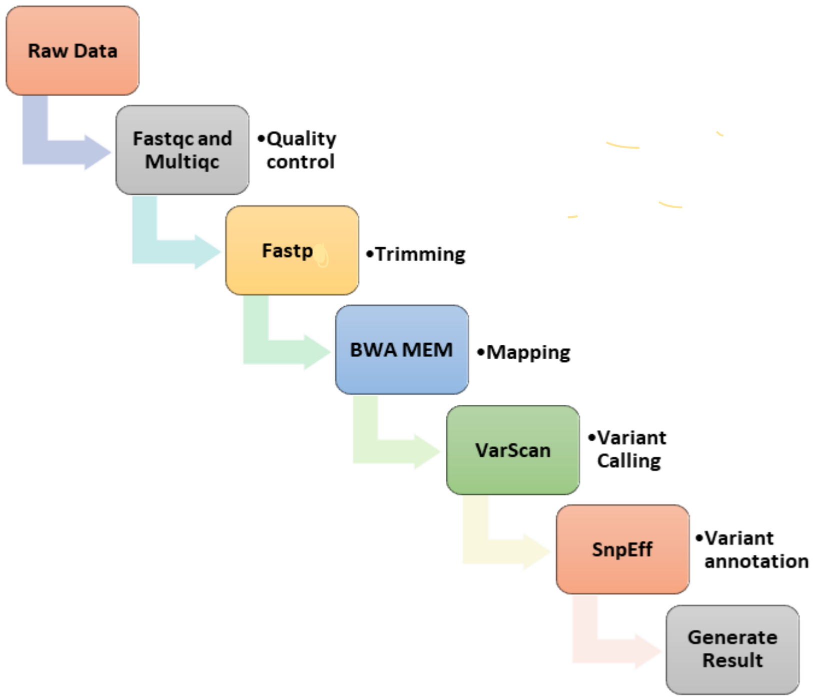

Fig 1. Graphical Abstract for the Task

In this task, we produced a Linux pipeline followed by Galaxy workflow to investigate the variant sites and the affected genes by identifying germline variants, somatic variants and variants affected by LOH from the tumor and normal tissue samples of a cancer patient.

I. LINUX PIPELINE

2.0 DATASETS

The datasets used in this analysis were obtained from a cancer patient's tumor as well as normal tissue samples **(Table 1)**.

Table 1. Datasets of normal and tumor samples

SAMPLES	TYPE
SLGFSK-N_231335_r1_chr5_12_17	Normal
SLGFSK-N_231335_r2_chr5_12_17	Normal
SLGFSK-T_231336_r1_chr5_12_17	Tumor
SLGFSK-T_231336_r2_chr5_12_17	Tumor

i) Downloading the datasets

Total of 4 datasets containing 2 normal and 2 tumor samples were downloaded from Zenodo using the wget command.

#Making directories

```
mkdir stage_two && cd stage_two
```

```
mkdir raw_data && cd raw_data
```

#Downloading sample datasets

```
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
```

#Downloading the reference sequence

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

#unzip the reference sequence

```
gunzip hg19.chr5_12_17.fa.gz
```

3.0 PRE-PROCESSING

i) Quality Control

Quality control helps in assessing the quality of the raw data. In this task, we have implemented FastQC and MultiQC on all the four raw datasets. FastQC carries out analysis on a single readsets, usually represented by a single fastq or fastq.gz file whereas MultiQC tool works directly on FastQC reports to generate a summary report for all processed samples.

#implementing fastQC

```
mkdir qc_reports
```

#implementing fastQC for all raw data

```
fastqc *.fastq.gz -o qc_reports/
```

#Implementing multiQC (assembling quality control reports)

```
multiqc qc_reports -o qc_reports
```

ii) Trimming

All the reads were trimmed using fastp. Fastp is an ultra fast tool used to perform quality control, adaptor trimming and quality filtering of datasets. The codes for implementing fastp can be found here

#implementing fastqc of trimmed_reads

```
cd \trimmed_reads
mkdir trimmed_qc_reports
fastqc *.fastq.gz -o trimmed_qc_reports/
```

#implementing multiqc of trimmed_reads

```
multiqc trimmed_qc_reports -o trimmed_qc_reports/
```

4.0 POST-PROCESSING

i) Mapping

The latest algorithm of BWA (Burrows-Wheeler Aligner), which is BWA-MEM, was implemented for mapping the reads. BWA is a fast short read aligner which is used for high quality queries as it is faster and more accurate. It is generally used for mapping low divergent sequences against a large reference genome, such as the human genome.

#Performing alignment

```
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 > mapping/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 > mapping/SLGFSK-T_231336.sam
```

The output generated by bwa-mem were two SAM files of the normal and tumor samples. SAM file or Sequence Alignment Map file is a type of text file format that is commonly used to store the results of nucleotide sequence alignments. Since they are in text file format, they are more readable by humans.

The SAM files were then converted to BAM files using samtools to generate more compressed files. BAM file or Binary Alignment Map is the compressed, binary version of a SAM file. BAM files are not readable by humans. BAM files are smaller and more efficient for the software to work with than the SAM file.

```
#converting sam to bam using samtools
samtools view -S -b SLGFSK-N_231335.sam > SLGFSK-N_231335.bam
samtools view -S -b SLGFSK-T_231336.sam > SLGFSK-T_231336.bam
```

After converting the SAM files to BAM files, the BAM files were sorted and indexed. This is because BAM files are large and they need to be indexed to perform computationally complex operations.

```
#sorting the bam files using samtools sort
samtools sort SLGFSK-N_231335.bam -o SLGFSK-N_231335.sorted.bam
samtools sort SLGFSK-T_231336.bam -o SLGFSK-T_231336.sorted.bam

#index sorted bam files using samtools index
samtools index SLGFSK-N_231335.sorted.bam
samtools index SLGFSK-T_231336.sorted.bam

#mapped reads filtering
samtools view -q 1 -f 0x2 -F 0x8 -b SLGFSK-N_231335.sorted.bam > SLGFSK-N_231335.filtered1.bam
samtools view -q 1 -f 0x2 -F 0x8 -b SLGFSK-T_231336.sorted.bam > SLGFSK-T_231336.filtered1.bam

#To view the output of the results use
samtools flagstat <bam file>
```

The next step was to carry out duplicates removal. Collate in samtools was used to shuffle and group reads together by their names. Then, fixmate in samtools was used to fill in mate coordinates and insert size fields. Further, markdup was used to mark duplicates and coordinate sorted files.

```
#This first collate command can be omitted if the file is already name ordered or collated
samtools collate SLGFSK-N_231335.filtered1.bam SLGFSK-N_231335.namecollate
samtools collate SLGFSK-T_231336.filtered1.bam SLGFSK-T_231336.namecollate
```

#Fixmate add ms and MC tags for markdup to use later

```
samtools fixmate -m SLGFSK-N_231335.namecollate.bam SLGFSK-N_231335.fixmate.bam
samtools fixmate -m SLGFSK-T_231336.namecollate.bam SLGFSK-T_231336.fixmate.bam
```

```
samtools sort -n -o SLGFSK-N_231335.namecollate.bam SLGFSK-N_231335.namecollate.bam
samtools sort -n -o SLGFSK-T_231336.namecollate.bam SLGFSK-T_231336.namecollate.bam
```

# sorting the files: This is because Markdup needs position order

```
samtools sort -@ 32 SLGFSK-N_231335.fixmate.bam -o SLGFSK-N_231335.positionsort.bam
samtools sort -@ 32 SLGFSK-T_231336.fixmate.bam -o SLGFSK-T_231336.positionsort.bam
```

```
samtools markdup -@32 -r SLGFSK-N_231335.positionsort.bam SLGFSK-N_231335.clean.bam
samtools markdup -@32 -r SLGFSK-T_231336.positionsort.bam SLGFSK-T_231336.clean.bam
```

```
cd ..
```

After removing the duplicates, the insertions and deletions in all the alignments the bam files were left-aligned and merged. Then, the samtools calmd was implemented to add MD and NM tags. Lastly, the bamtools filter was used to refilter the read mapping qualities.

#left align bam

```
cat mapping/SLGFSK-N_231335.clean.bam  | bamleftalign -f reference/hg19.chr5_12_17.fa -m 5 -c > mapping/SLGFSK-N_231335.leftAlign.bam
```

```
cat mapping/SLGFSK-T_231336.clean.bam  | bamleftalign -f reference/hg19.chr5_12_17.fa -m 5 -c > mapping/SLGFSK-T_231336.leftAlign.bam
```

#Recalibrate read mapping qualities

```
samtools calmd -@ 32 -b mapping/SLGFSK-N_231335.leftAlign.bam reference/hg19.chr5_12_17.fa > mapping/SLGFSK-N_231335.recalibrate.bam
```

```
samtools calmd -@ 32 -b mapping/SLGFSK-T_231336.leftAlign.bam reference/hg19.chr5_12_17.fa > mapping/SLGFSK-T_231336.recalibrate.bam
```

#Refilter read mapping qualities

```
bamtools filter -in mapping/SLGFSK-N_231335.recalibrate.bam -mapQuality "<=254" > mapping/SLGFSK-N_231335.refilter.bam
```

```
bamtools filter -in mapping/SLGFSK-T_231336.recalibrate.bam -mapQuality "<=254" > mapping/SLGFSK-T_231336.refilter.bam
```

ii. Variant calling and classification

Variant calling is the process of identifying variants from sequence data. It creates a way to store genetic variation data between sample and reference in a readable and user friendly file called VCF files. To identify variants from the mapped samples, the tool VarScan somatic was used. VarScan is an open source tool for detecting SNPs, insertions, and deletions and assessing their frequencies in massively parallel sequencing data. Given an alignments file, VarScan scores and sorts the alignments on a per-read basis, discarding reads that are aligned with low identity or to multiple locations in the reference sequence.

$$\begin{tikzcd}
\textbf{FASTQ files} \arrow[r, Rightarrow] & \textbf{BAM files/CRAM} \arrow[r, Rightarrow] & \textbf{VCF files} \\
\textbf{\textit{(Whole genome sequencing)}} \arrow[u] & \textbf{\textit{(Alignment)}} \arrow[u] & \textbf{\textit{(Extract Variants between samples)}} \arrow[u]
\end{tikzcd}$$

The VarScan somatic tool requires samples in samtools pileup format. Pileup format is a text-based format for summarizing the base calls of aligned reads to a reference sequence. This format facilitates visual display of SNP/indel calling and alignment.

#Convert data to pileup using samtools mpileup

```
mkdir variants
```

```
samtools mpileup -f reference/hg19.chr5_12_17.fa mapping/SLGFSK-N_231335.refilter.bam --min-MQ 1 --min-BQ 28 > variants/SLGFSK-N_231335.pileup
```

```
samtools mpileup -f reference/hg19.chr5_12_17.fa mapping/SLGFSK-T_231336.refilter.bam --min-MQ 1 --min-BQ 28 > variants/SLGFSK-T_231336.pileup
```

After converting the samples pileup format, VarScan somatic was implemented which generated SNP and Indels vcf file. Next, the single best alignment for each read was screened for sequence changes. Variants detected in multiple reads are then combined together into unique SNPs and indels. VCF (Variant Call Format) is a text file format which contains meta-information lines, a header line and data lines containing information about a position in the genome. It can also contain genotype information on samples for each position.

```
#Call variants
Java -jar VarScan.v2.3.9.jar somatic variants/SLGFSK-N_231335.pileup variants/SLGFSK-
T_231336.pileup variants/SLGFSK \ --normal-purity 1 --tumor-purity 0.5 --output-vcf 1
```

In order to sort position and compress the input data vcf files before indexing, bgzip was used which has a gzip-like interface. This was followed by indexing which was done using tabix. Tabix allows indexes to be built against the compressed file and used to retrieve portions of the data without having to decompress the entire file. It indexes a TAB-delimited genome position file and creates an index file. It is used so that it will be easier and faster to retrieve data lines from overlapping regions. Next, bcftools merge was used to merge the indexed of files which generated a single vcf file.

```
#merge vcf
bgzip variants/SLGFSK.snp.vcf > variants/SLGFSK.snp.vcf.gz
bgzip variants/SLGFSK.indel.vcf > variants/SLGFSK.indel.vcf.gz
tabix variants/SLGFSK.snp.vcf.gz
tabix variants/SLGFSK.indel.vcf.gz
cd variants
#merge vcf files with bcftools
bcftools merge SLGFSK.snp.vcf.gz SLGFSK.indel.vcf.gz -o SLGFSK.vcf --force -samples
```

The complete LINUX pipeline can be found here
Note: The GEMINI annotation requires the additional download of annotation resources and databases which requires high storage space. Hence, the generated VCF file was uploaded to Galaxy to implement the annotation steps.

II. GALAXY WORKFLOW

5.0 VARIANT ANNOTATION AND REPORTING

Variant annotation is the process of assigning information to DNA variants. It is a crucial step in linking sequence variants with changes in phenotype.

In this section, we used gene and variant annotations from different sources released either in the public domain or under a free data license:

i. Cancer Hotspots
ii. Cancer Biomarkers Database of the Cancer Genome Interpreter Project (CGI)
iii. CIVic database
iv. dbSNP
v. Uniprot

The *"Upload Data"* option in the galaxy was used to copy and paste the variant annotation file links from Zenodo in which the datatype selected for the first three files was "*bed"* and for the last file was "*vcf"*.

https://zenodo.org/record/2581873/files/hotspots.bed
https://zenodo.org/record/2581873/files/cgi_variant_positions.bed
https://zenodo.org/record/2581873/files/01-Feb-2019-CIVic.bed
https://zenodo.org/record/2582555/files/dbsnp.b147.chr5_12_17.vcf

Similarly, gene-level annotation files were also from Zenodo with "*tabular"* as their datatype.

https://zenodo.org/record/2581881/files/Uniprot_Cancer_Genes.13Feb2019.txt
https://zenodo.org/record/2581881/files/cgi_genes.txt
https://zenodo.org/record/2581881/files/01-Feb-2019-GeneSummaries.tsv

i. Functional Annotation using SnpEff

No variants are equal so, it is important to know which genes are affected by the variants. For this, SnpEff is a useful tool. It is a genetic variant annotation and functional effect prediction toolbox. It annotates and predicts the effects of genetic variants on genes and proteins (such as amino acid changes).
The merged vcf file generated from VarScan in the last step of the Linux pipeline was used as an input.

SnpEff was run using the following parameters:

- ❖ Sequence changes (SNPs, MNPs, InDels): the output of the VarScan somatic tool
- ❖ Input format: VCF
- ❖ Output format: VCF (only if the input is VCF)
- ❖ Genome source: Locally installed reference genome
- ❖ Genome: Homo sapiens: hg19 (or a similarly named option)

ii. Clinical and genetic annotation using GEMINI

Other interesting information about the variants could be in how much frequency it has been observed in a human population or whether it is known to be associated with diseases. To proceed with this kind of genetic and clinical annotations GEMINI was used. GEMINI (GEnome MINIng) is a flexible framework for exploring genetic variation.

The output annotation file produced by SnpEff was used as input "*Gemini load*" into the GEMINI database with the optional contents: GERP scores, CADD scores, Gene tables, Sample genotypes and Variant INFO field. Further annotations to the variants already loaded in the GEMINI database were added using the Gemini Annotate tool. This tool was used to add more explicit information on the output of VarScan that could not be identified by "*Gemini load*" tool. The values of Somatic status (SS), Germline p-value (GPV), and Somatic p-value (SPV) extracted from the VarScan for each variant detected were added to the GEMINI database. Additional annotations were added from dbSNP, Cancer hotspots, links to the CIVic database and CGI using the "*Gemini Annotate*" tool.

iii. Reporting selected subsets of variants using GEMINI query

After building and enriching the GEMINI database with additional annotations, filtered variant reports listing high-quality variants (somatic, germline, LOH) were achieved using the steps in the *"GEMINI query"*. Gemini query is used for querying a GEMINI database that has been loaded using the "*GEMINI load*" command. The query syntax is built on the SQLite dialect of SQL. Further, more complex filters with additional constraints and comma-separated columns were also added.

iv. Generating reports of genes affected by variants

In this step, a gene-centred report was generated based on the same somatic variants that were selected above. In this report, the annotations were applied to the whole gene affected by a variant rather than to the variant itself. Here, the "*GEMINI query*" was built using "*Advanced query constructor*" and the query was issued to the database. However, the "*Genotype filter expression*" remained the same.

v. Adding additional annotations to the gene-centred report

The addition of extra annotations to the *GEMINI* - generated gene report was to make the interpretation of the final output easier. Since the *GEMINI* annotate didn't allow the inclusion of additional annotations into the tabular gene report, the task was achieved using the general-purpose Galaxy tools. This helped in the removal of irrelevant columns by specifying and rearranging the remaining ones.

Three step-wise processes were involved here:

1) Pulled the annotations found in the Uniprot cancer genes dataset.
2) Annotated the newly formed gene-centred report with the CGI biomarkers datasets, by using the output of the last Join operation.
3) Added the Gene Summaries dataset by using the output of the second Join operation.

Lastly, the *"Column arrange"* by header name was run to rearrange the fully-annotated gene-centred report and eliminate unspecified columns. The last output of the Join operation was selected in the *"file to arrange"* section. The columns were specified by name and the result was a tabular gene report which was easy to understand and interpret.

The complete GALAXY workflow can be found here

6.0 RESULTS

i) Quality Control

Almost all the parameters remained unchanged after the quality control of the raw reads before and after trimming. However, all the samples had the same sequence length before trimming but, the sequence length distribution varied after the quality control of the trimmed reads. Distribution of fragment sizes were found. Also, no adapter contamination of >0.1% was found after trimming. The summarized multiqc results of the raw reads before and after trimming can be found in the GitHub repository.

ii) Annotation

The tabular gene report of the additional annotations which were not possible using GEMINI-annotate was generated using the "*Join two files"* tools on Galaxy. Irrelevant and unspecified columns were removed by specifying the columns required. Further, Column arrange was run by header name to rearrange and produce the fully annotated gene-centred report. Lastly, the tabular gene report was generated with the last output of the Join operation which was easy to understand and interpret. Around 42 affected genes were reported in chromosomes 5, 17 and 12 which included commonly affected genes such as KRAS and TP53. Few of the genes were also found to be associated with biomarkers of cancer subtypes as well as other diseases.

7.0 CONCLUSION

Somatic variant calling demonstrates an excellent method to distinguish the somatic mutations (confined/specific to the tumor tissue) from the germline mutations (that are shared by both tumor and healthy tissue) and LOH events (which involve loss of one or two alleles from tumor tissue). Hence, it proves to be an optimal approach rather than just merely calling the variants. However, the interpretation of any list of variants (somatic, germline or LOH) depends on rich genetic and cancer-specific variants and gene annotations.

8.0 REFERENCE MATERIALS

1. https://training.galaxyproject.org/training-material/topics/variant-analysis/tutorials/somatic-variants/tutorial.html
2. https://github.com/Fredrick-Kakembo/Somatic-and-Germline-variant-Identification-from-Tumor-and-normal-Sample-Pairs/blob/main/README.md