Workshop 2024 commands

- Open MobaXterm ("Click on the Windows icon on the lower left corner, search for MobaXterm, click on the MobaXterm icon") Type the follwing command to login to your account (username@ipaddress)

```
ssh ciw12@10.100.75.81
```

- Go to the terminal and type `pwd`.
- Make sure you are in `/scratch/ciw/ciw01` (ciw01 to ciw30 depending on your user)

Make a directory for output

- Now, type `mkdir output` into the terminal.
- Type `ls` and press `Enter`. You should be able to see 3 files `commands.log`, `path_files.dat`, `run_script.sh` and 1 directory `output` which you just created.

0. Quality Control

- Now open the `run_script.sh` file from the left hand side panel of your screen. (Right click on it and select `Open with default text editor`)
- You will see something like this.

```
#!/usr/bin/bash
#BSUB -J example        # Job name
#BSUB -n 1          # No. of threads
#BSUB -o output/output.log  # Output file for standard output
#BSUB -e output/error.log   # Output file for standard error

source path_files.dat
```

- Don't change anything. Just copy and paste the following command onto the new line below source path_files.dat

```
fastqc ${sequences}/24NGS775-B1_S55_R1_001.fastq.gz ${sequences}/24NGS775-
B1_S55_R2_001.fastq.gz -o output/
```

- Now your file should look like this

```
#!/usr/bin/bash
#BSUB -J example        # Job name
#BSUB -n 1          # No. of threads
#BSUB -o output/output.log  # Output file for standard output
```

```
#BSUB -e output/error.log   # Output file for standard error

source path_files.dat

fastqc ${sequences}/24NGS775-B1_S55_R1_001.fastq.gz ${sequences}/24NGS775-
B1_S55_R2_001.fastq.gz -o output/
```

- Now press `Ctr + S` to save the file. Close it.
- Wait for the progress bar on the lower left hand corner of your screen to finish.
- Then type the following command to view the script on the terminal to make sure you have modified it correctly.

```
cat run_script.sh
```

- This will print the whole script on terminal itself. Check the script again.
- Now run the script by typing the following onto the terminal.

```
bsub < run_script.sh
```

bsub is a command used to submit jobs to the LSF (Load Sharing Facility) job scheduler. LSF is a workload management platform used in high-performance computing (HPC) environments to manage, schedule, and distribute computing tasks across a cluster of machines. When you use bsub, you're asking the system to submit a job to the LSF queue, where it will be scheduled and run on an available node (a machine in the cluster).

- Once, you submit the job by typing `bsub < run_script.sh`, you can check it's status by simply typing

```
bjobs
```

You will see one of the following on the terminal.

```
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
64141   ciw03   RUN   normal     hpc         cn3         example    Sep 19 09:53
```

OR

```
No unfinished job found
```

- If you see the first output, that means your job is still running and you should wait for it to finish. Type `bsub` again after some time.

- If you see the second output `No unfinished job found`, it means your job is finished.
- You can check the output of the command you just ran by typing

```
ls -ltrh output/
```

Similarly, do this for every step below and check if the output files are generated by typing `ls -ltrh output/`

We'll do it for one more step.

- Open the `run_script.sh` with default text editor and remove the last command you copy pasted into the script which was `fastqc`.
- Now, your script will look like this.

```
#!/usr/bin/bash
#BSUB -J example        # Job name
#BSUB -n 1          # No. of threads
#BSUB -o output/output.log  # Output file for standard output
#BSUB -e output/error.log   # Output file for standard error

source path_files.dat
```

1. Trimming

- Now, copy paste the following command as it is onto the new line below `source path_files.dat` like you did earlier.

```
trimmomatic PE ${sequences}/24NGS775-B1_S55_R1_001.fastq.gz ${sequences}/24NGS775-
B1_S55_R2_001.fastq.gz -baseout output/24NGS775-B1.fq.gz
ILLUMINACLIP:$adapters:2:30:10:2:keepBothReads LEADING:3 SLIDINGWINDOW:4:15
MINLEN:40
```

- Your script should look like this.

```
#!/usr/bin/bash
#BSUB -J example        # Job name
#BSUB -n 1          # No. of threads
#BSUB -o output/output.log  # Output file for standard output
#BSUB -e output/error.log   # Output file for standard error

source path_files.dat

trimmomatic PE ${sequences}/24NGS775-B1_S55_R1_001.fastq.gz ${sequences}/24NGS775-
B1_S55_R2_001.fastq.gz -baseout output/24NGS775-B1.fq.gz
```

```
ILLUMINACLIP:$adapters:2:30:10:2:keepBothReads LEADING:3 SLIDINGWINDOW:4:15
MINLEN:40
```

- Now press `Ctr + S` to save the file. Close it.
- Wait for the progress bar on the lower left hand corner of your screen to finish.
- Then type the following command to view the script on the terminal to make sure you have modified it correctly.

```
cat run_script.sh
```

- Check the modified script again.
- Now, Run the script by typing

```
bsub < run_script.sh
```

Check the job's status by typing

```
bjobs
```

If it is finished, check the output files in the `output` directory by typing

```
ls -ltrh output/
```

Do this for every step.

2. Paired end assembly

```
flash output/24NGS775-B1_1P.fq.gz output/24NGS775-B1_2P.fq.gz --cap-mismatch-quals
-O -M 250 -o output/24NGS775-B1
```

3. Mapping to reference genome

```
bwa mem -R "@RG\tID:AML\tPL:ILLUMINA\tLB:LIB-MIPS\tSM:24NGS775-B1\tPI:200" -M -t
20 ${genome} output/24NGS775-B1.extendedFrags.fastq > output/24NGS775-B1.sam
```

4. Sam conversion

```
samtools view -b output/24NGS775-B1.sam > output/24NGS775-B1.bam
samtools sort output/24NGS775-B1.bam > output/24NGS775-B1.sorted.bam
samtools index output/24NGS775-B1.sorted.bam > output/24NGS775-B1.sorted.bam.bai
```

5. GATK Best Practices for data pre-processing

Details of this step can be found here : https://gatk.broadinstitute.org/hc/en-us/articles/360035535912-Data-pre-processing-for-variant-discovery

```
java -Xmx8G -jar ${GATK38_path} -T RealignerTargetCreator -R ${genome} -nt 10 -I
output/24NGS775-B1.sorted.bam --known ${site1} -o output/24NGS775-B1.intervals

java -Xmx8G -jar ${GATK38_path} -T IndelRealigner -R ${genome} -I output/24NGS775-
B1.sorted.bam -known ${site1} --targetIntervals output/24NGS775-B1.intervals -o
output/24NGS775-B1.realigned.bam

java -Xmx8G -jar ${GATK38_path} -T BaseRecalibrator -R ${genome} -I
output/24NGS775-B1.realigned.bam -knownSites ${site2} -knownSites ${site3} -
maxCycle 600 -o output/24NGS775-B1.recal_data.table

java -Xmx8G -jar ${GATK38_path} -T PrintReads -R ${genome} -I output/24NGS775-
B1.realigned.bam --BQSR output/24NGS775-B1.recal_data.table -o output/24NGS775-
B1.final.bam
```

6. Coverage calculation

```
bedtools bamtobed -i output/24NGS775-B1.final.bam > output/24NGS775-B1.final.bed
bedtools coverage -counts -a ${bedfile}.bed -b output/24NGS775-B1.final.bed >
output/24NGS775-B1.counts.bed
```

7. Variant calling

```
java -Xmx10G -jar ${GATK38_path} -T MuTect2 -R ${genome} -I:tumor output/24NGS775-
B1.final.bam -o output/24NGS775-B1_mutect.vcf -L ${bedfile}.bed
```

8. Variant annotation

```
convert2annovar.pl -format vcf4 output/24NGS775-B1_vardict.vcf --outfile
output/24NGS775-B1.avinput --withzyg --includeinfo

table_annovar.pl output/24NGS775-B1.avinput --out output/24NGS775-B1_final --
remove --protocol refGene,cosmic84,exac03 --operation g,f,f --buildver hg19 --
nastring '-1' --otherinfo --csvout ${database}
```

9. Format output

```
python3 ${formatMutect_script_path} output/24NGS775-B1_final.hg19_multianno.csv
24NGS775-B1 output/
```

10. KDMdb

```
python3 ${KDMdb_script_path} output/24NGS775-B1_mutect.csv output/ 24NGS775-B1
```