Project #2. “Why did I get the flu?”. Deep sequencing, error control, p-value, viral evolution”

Tags

Made by Maria Uzun and Alisa Fedorenko

1. Get the data

Create a directory for Project 2 materials, and inside it create a new directory for
raw data:

```
mkdir Project2
cd Project2/
mkdir raw_data
cd raw_data
```

Download the roommate’s data. It was published there and labeled SRR1705851, so you can download it from the SRA FTP server and unpack it on your machine:

```
wget http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/001/SRR1705851/SRR1705851.fastq.gz
```

Download the reference: tap here (just copy and paste, or select Send to -> File -> FASTA format). Then copy it in your working directory raw_data folder.

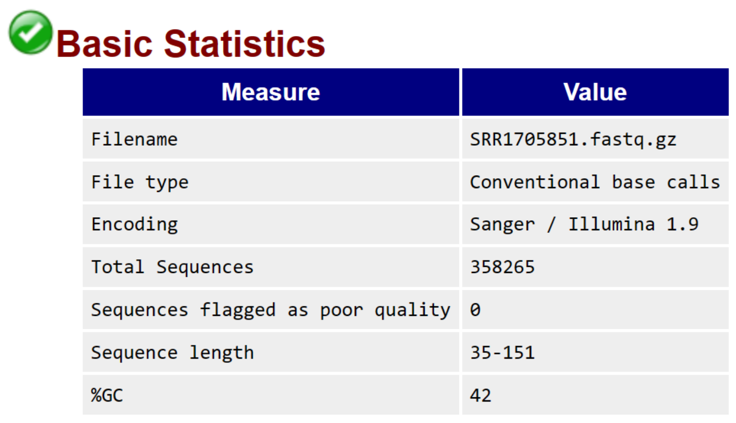
We downloaded sequence.fasta file. Let’s rename it:

```
mv sequence.fasta influenza_hemagglutinin_gene.fasta
```

2. Inspect raw sequencing data with FastQC.

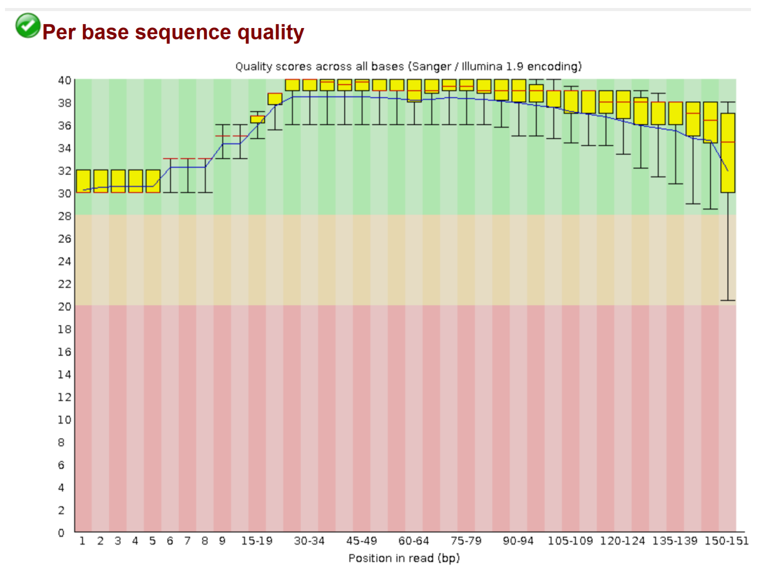
Turn back to your project folder and create fastqc folder to check the quality of the reads:

```
cd ..
mkdir fastqc
cd fastqc
fastqc -o ./ ../raw_data/SRR1705851.fastq.gz
```



Basic Statistics

Measure	Value
Filename	SRR1705851.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	358265
Sequences flagged as poor quality	0
Sequence length	35-151
%GC	42



We have good quality of reads, so we do not need to trim our reads.

3. Index reference genome.

3.1 Index the reference file

Run bwa index on the reference sequence with the default options

```
cd ..
mkdir alignment
cd alignment
cp ../raw_data/influenza_hemagglutinin_gene.fasta ./
#conda install -c bioconda bwa
bwa index ./influenza_hemagglutinin_gene.fasta
```

You will see:

```
[bwa_index] Pack FASTA... 0.00 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 0.00 seconds elapse.
[bwa_index] Update BWT... 0.00 sec
[bwa_index] Pack forward-only FASTA... 0.00 sec
[bwa_index] Construct SA from BWT and Occ... 0.00 sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa index ./influenza_hemagglutinin_gene.fasta
[main] Real time: 0.057 sec; CPU: 0.005 sec
```

You need to see the several files after the indexing:

```
influenza_hemagglutinin_gene.fasta
influenza_hemagglutinin_gene.fasta.amb
influenza_hemagglutinin_gene.fasta.ann
influenza_hemagglutinin_gene.fasta.bwt
influenza_hemagglutinin_gene.fasta.pac
influenza_hemagglutinin_gene.fasta.sa
```

3.2 Aligning reads

```
bwa mem ./influenza_hemagglutinin_gene.fasta ../raw_data/SRR1705851.fastq.gz | \
samtools view -S -b - | samtools sort -o alignment_SRR1705851_sorted.bam -
```

3.3 Check flagstat statistics

```
samtools flagstat alignment_SRR1705851_sorted.bam
```

You will see the results like this:

```
361349 + 0 in total (QC-passed reads + QC-failed reads)
358265 + 0 primary
0 + 0 secondary
3084 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
361116 + 0 mapped (99.94% : N/A)
358032 + 0 primary mapped (99.93% : N/A)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

3.4 Sort and index BAM file

a) Index bam file for faster search:

```
samtools index alignment_SRR1705851_sorted.bam
```

After all procedures you will see this list of the files:

```
alignment_SRR1705851_sorted.bam
alignment_SRR1705851_sorted.bam.bai
influenza_hemagglutinin_gene.fasta
influenza_hemagglutinin_gene.fasta.amb
influenza_hemagglutinin_gene.fasta.ann
influenza_hemagglutinin_gene.fasta.bwt
influenza_hemagglutinin_gene.fasta.pac
influenza_hemagglutinin_gene.fasta.sa
```

4. Variant calling

The solution is to make an intermediate file type called an mpileup, because it goes through each position and “piles up” the reads, tabulating the number of bases that match or don’t match the reference. Mpileup requires a sorted, indexed bam file. For this, run the basic command below.

```
cd ..
mkdir variant_calling
```

```
cd variant_calling
samtools mpileup -f ../alignment/influenza_hemagglutinin_gene.fasta ../alignment/alignment_SRR1705851_sorted.bam -d 0 > my.mpileup
```

To call actual variants, we will be using a program called VarScan (variant scanner)

T*he desired value of d should be* ***0*** *if we want to keep all possible variants, given the length of the reference sequence, the number of reads, and the average read length.*

To call actual variants, we will be using a program called VarScan (variant scanner)

Download version 2.4.0 to the variant_calling directory from here https://github.com/dkoboldt/varscan/blob/master/VarScan.v2.4.0.jar

4.1 Looking for the common variants with VarScan

For SNPs checking run the following command. We set a high minimum variant frequency cut-off (N) to find only those mutants present in most (95% or more = 0.95) of the viral DNA molecules

```
java -jar VarScan.v2.4.0.jar mpileup2snp my.mpileup --min-var-freq 0.95 --variants --output-vcf 1 > VarScan_results.vcf
```

5 variant positions reported (5 SNP, 0 indel)

```
Only SNPs will be reported
Warning: No p-value threshold provided, so p-values will not be calculated
Min coverage:   8
Min reads2:     2
Min var freq:   0.95
Min avg qual:   15
P-value thresh: 0.01
Reading input from my.mpileup
1665 bases in pileup file
5 variant positions (5 SNP, 0 indel)
0 were failed by the strand-filter
5 variant positions reported (5 SNP, 0 indel)
```

Let’s see the results. Awk helps us to parse the document to extract needed information:

```
cat VarScan_results.vcf | awk 'NR>24 {print $1, $2, $4, $5}'
```

NR stands for number of read lines, so it’s saying ‘only if the line number is >24, do what’s next’.

Awk uses the dollar sign to mark fields (columns), so what’s next is ‘print columns 1,2, 4 and 5”.

The output:

```
KF848938.1 72 A G
KF848938.1 117 C T
KF848938.1 774 T C
KF848938.1 999 C T
KF848938.1 1260 A C
```

All of these mutations are synonimous and do not affect the andibodies binding to epitope.

4.2 Look for rare variants with VarScan

Set the minimum variant frequency to 0.001 (0.1%) and run the scan again on the same mpileup file.

```
java -jar VarScan.v2.4.0.jar mpileup2snp my.mpileup --min-var-freq 0.001 --variants --output-vcf 1 > VarScan_results_0001.vcf
cat VarScan_results_0001.vcf | awk 'NR>24 {match($10, /([0-9.]+)%/, m); print $1, $2, $4, $5, m[1]}' > snps_0001.txt
```

All the found mutations listed below in the table.

5. Inspect and align the control sample sequencing data

Download fastq data for the three controls (from sequencing of isogenic reference samples) from SRA FTP:

```
cd ../raw_data
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/008/SRR1705858/SRR1705858.fastq.gz
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/009/SRR1705859/SRR1705859.fastq.gz
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/000/SRR1705860/SRR1705860.fastq.gz
```

How many reads are in each file

```
seqkit stats SRR1705858.fastq.gz
```

File	Format	Type	Num_seqs	Sum_len	Min_len	Avg_len	Max_len
SRR1705858.fastq.gz	FASTQ	DNA	256,586	38,118,730	35	148.6	151

```
seqkit stats SRR1705859.fastq.gz
```

File	Format	Type	Num_seqs	Sum_len	Min_len	Avg_len	Max_len
SRR1705859.fastq.gz	FASTQ	DNA	233,327	34,636,567	35	148.4	151

```
seqkit stats SRR1705860.fastq.gz
```

File	Format	Type	Num_seqs	Sum_len	Min_len	Avg_len	Max_len
SRR1705860.fastq.gz	FASTQ	DNA	249,964	37,170,486	35	148.7	151

Align each control fastq file to the reference (KF848938.1.fasta), convert it to a bam file, and sort it. You need to make a separate bam alignment for each control sample, but you do not have to index the reference sequence again. Be sure to give each bam file a unique name. Use samtools to index each of the new control alignment (bam) files.

```
cd ../alignment
bwa mem ./influenza_hemagglutinin_gene.fasta ../raw_data/SRR1705858.fastq.gz | \
samtools view -S -b - | samtools sort -o alignment_SRR1705858_sorted.bam -
samtools index alignment_SRR1705858_sorted.bam

bwa mem ./influenza_hemagglutinin_gene.fasta ../raw_data/SRR1705859.fastq.gz | \
samtools view -S -b - | samtools sort -o alignment_SRR1705859_sorted.bam -
samtools index alignment_SRR1705859_sorted.bam

bwa mem ./influenza_hemagglutinin_gene.fasta ../raw_data/SRR1705860.fastq.gz | \
samtools view -S -b - | samtools sort -o alignment_SRR1705860_sorted.bam -
samtools index alignment_SRR1705860_sorted.bam
```

As a result, there are bam, sorted.bam and bai files for each control variants

```
samtools flagstat alignment_SRR1705858_sorted.bam
samtools flagstat alignment_SRR1705859_sorted.bam
samtools flagstat alignment_SRR1705860_sorted.bam
```

For alignment_SRR1705858_sorted.bam:

```
256744 + 0 in total (QC-passed reads + QC-failed reads)
256586 + 0 primary
0 + 0 secondary
158 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
256658 + 0 mapped (99.97% : N/A)
256500 + 0 primary mapped (99.97% : N/A)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

For alignment_SRR1705859_sorted.bam:

```
233451 + 0 in total (QC-passed reads + QC-failed reads)
233327 + 0 primary
0 + 0 secondary
124 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
233375 + 0 mapped (99.97% : N/A)
233251 + 0 primary mapped (99.97% : N/A)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

For alignment_SRR1705860_sorted.bam:

```
250184 + 0 in total (QC-passed reads + QC-failed reads)
249964 + 0 primary
0 + 0 secondary
220 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
250108 + 0 mapped (99.97% : N/A)
249888 + 0 primary mapped (99.97% : N/A)
```

0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)

6. Use VarScan to look for rare variants in the reference files.

Run VarScan with a minimum variant frequency of 0.001 (0.1%) on each of the reference alignments. Be sure to tell VarScan to only output variants and to format the output in the vcf format.

```
cd ../variant_calling
samtools mpileup -f ../alignment/influenza_hemagglutinin_gene.fasta ../alignment/alignment_SRR1705858_sorted.bam -d 0 > my_SRR1705858_sorted.mpileup
java -jar VarScan.v2.4.0.jar mpileup2snp my_SRR1705858_sorted.mpileup --min-var-freq 0.001 --variants --output-vcf 1 > VarScan_results_0001_SRR1705858.vcf

samtools mpileup -f ../alignment/influenza_hemagglutinin_gene.fasta ../alignment/alignment_SRR1705859_sorted.bam -d 0 > my_SRR1705859_sorted.mpileup
java -jar VarScan.v2.4.0.jar mpileup2snp my_SRR1705859_sorted.mpileup --min-var-freq 0.001 --variants --output-vcf 1 > VarScan_results_0001_SRR1705859.vcf

samtools mpileup -f ../alignment/influenza_hemagglutinin_gene.fasta ../alignment/alignment_SRR1705860_sorted.bam -d 0 > my_SRR1705860_sorted.mpileup
java -jar VarScan.v2.4.0.jar mpileup2snp my_SRR1705860_sorted.mpileup --min-var-freq 0.001 --variants --output-vcf 1 > VarScan_results_0001_SRR1705860.vcf
```

Parse each vcf file so that you get three lists containing the reference base, position, alternative base, and frequency. Copy those lists into a spreadsheet of your choice (e.g. Excel, Google Sheets, R, etc.) and, possibly, into your lab notebook

```
cat VarScan_results_0001_SRR1705858.vcf | awk 'NR>24 {match($10, /([0-9.]+)%/, m); print $1, $2, $4, $5, m[1]}' > snps_SRR1705858.txt
cat VarScan_results_0001_SRR1705859.vcf | awk 'NR>24 {match($10, /([0-9.]+)%/, m); print $1, $2, $4, $5, m[1]}' > snps_SRR1705859.txt
cat VarScan_results_0001_SRR1705860.vcf | awk 'NR>24 {match($10, /([0-9.]+)%/, m); print $1, $2, $4, $5, m[1]}' > snps_SRR1705860.txt
```

The obtained results are:

Roommate	Position	Reference base	Alternative base	Frequency	SRR1705858	Position	Reference base	Alternative base	Frequency
KF848938.1	72	A	G	99.96	KF848938.1	38	T	C	0.66
KF848938.1	117	C	T	99.82	KF848938.1	54	T	C	0.3
KF848938.1	254	A	G	0.17	KF848938.1	72	A	G	0.3
KF848938.1	276	A	G	0.17	KF848938.1	95	A	G	0.24
KF848938.1	307	C	T	0.94	KF848938.1	117	C	T	0.3
KF848938.1	340	T	C	0.17	KF848938.1	165	T	C	0.24
KF848938.1	389	T	C	0.22	KF848938.1	183	A	G	0.3
KF848938.1	691	A	G	0.17	KF848938.1	216	A	G	0.22
KF848938.1	722	A	G	0.2	KF848938.1	218	A	G	0.28
KF848938.1	744	A	G	0.17	KF848938.1	222	T	C	0.26
KF848938.1	774	T	C	99.96	KF848938.1	235	T	C	0.25
KF848938.1	802	A	G	0.23	KF848938.1	254	A	G	0.25
KF848938.1	859	A	G	0.18	KF848938.1	276	A	G	0.22
KF848938.1	915	T	C	0.19	KF848938.1	297	T	C	0.2
KF848938.1	999	C	T	99.86	KF848938.1	328	T	C	0.2
KF848938.1	1043	A	G	0.18	KF848938.1	340	T	C	0.23
KF848938.1	1086	A	G	0.21	KF848938.1	356	A	G	0.22
KF848938.1	1213	A	G	0.22	KF848938.1	370	A	G	0.21
KF848938.1	1260	A	C	99.94	KF848938.1	389	T	C	0.23
KF848938.1	1280	T	C	0.18	KF848938.1	409	T	C	0.22
KF848938.1	1458	T	C	0.84	KF848938.1	414	T	C	0.28
					KF848938.1	421	A	G	0.18
					KF848938.1	426	A	G	0.19
					KF848938.1	463	A	G	0.19
					KF848938.1	516	A	G	0.2
					KF848938.1	566	A	G	0.22
					KF848938.1	595	G	T	0.34
					KF848938.1	597	A	G	0.17
					KF848938.1	660	A	G	0.2
					KF848938.1	670	A	G	0.29
					KF848938.1	691	A	G	0.23
					KF848938.1	722	A	G	0.23
					KF848938.1	744	A	G	0.21
					KF848938.1	774	T	C	0.3

Roommate	Position	Reference base	Alternative base	Frequency	SRR1705858	Position	Reference base	Alternative base	Frequency
					KF848938.1	859	A	G	0.27
					KF848938.1	915	T	C	0.26
					KF848938.1	987	A	G	0.22
					KF848938.1	1008	T	G	0.27
					KF848938.1	1031	A	G	0.28
					KF848938.1	1043	A	G	0.24
					KF848938.1	1056	T	C	0.2
					KF848938.1	1086	A	G	0.33
					KF848938.1	1089	A	G	0.22
					KF848938.1	1213	A	G	0.24
					KF848938.1	1260	A	C	0.3
					KF848938.1	1264	T	C	0.26
					KF848938.1	1280	T	C	0.25
					KF848938.1	1281	T	C	0.22
					KF848938.1	1286	T	C	0.2
					KF848938.1	1339	T	C	0.41
					KF848938.1	1358	A	G	0.26
					KF848938.1	1398	T	C	0.2
					KF848938.1	1421	A	G	0.31
					KF848938.1	1460	A	G	0.34
					KF848938.1	1482	A	G	0.24
					KF848938.1	1580	T	C	0.25
					KF848938.1	1591	T	C	0.29
SRR1705859	Position	Reference base	Alternative base	Frequency	SRR1705860	Position	Reference base	Alternative base	Frequency
KF848938.1	44	T	C	0.47	KF848938.1	38	T	C	0.7
KF848938.1	158	A	G	0.24	KF848938.1	44	T	C	0.5
KF848938.1	165	T	C	0.27	KF848938.1	95	A	G	0.24
KF848938.1	183	A	G	0.22	KF848938.1	105	A	G	0.25
KF848938.1	193	A	G	0.22	KF848938.1	133	A	G	0.22
KF848938.1	216	A	G	0.24	KF848938.1	158	A	G	0.26
KF848938.1	218	A	G	0.29	KF848938.1	165	T	C	0.25
KF848938.1	222	T	C	0.25	KF848938.1	183	A	G	0.23
KF848938.1	254	A	G	0.19	KF848938.1	199	A	G	0.19
KF848938.1	276	A	G	0.24	KF848938.1	216	A	G	0.24
KF848938.1	319	T	C	0.23	KF848938.1	218	A	G	0.23
KF848938.1	340	T	C	0.21	KF848938.1	222	T	C	0.3
KF848938.1	356	A	G	0.24	KF848938.1	228	T	C	0.19
KF848938.1	370	A	G	0.21	KF848938.1	230	A	G	0.19
KF848938.1	398	A	G	0.22	KF848938.1	235	T	C	0.25
KF848938.1	403	A	G	0.19	KF848938.1	254	A	G	0.23
KF848938.1	409	T	C	0.19	KF848938.1	271	A	G	0.21
KF848938.1	414	T	C	0.22	KF848938.1	276	A	G	0.33
KF848938.1	421	A	G	0.18	KF848938.1	297	T	C	0.23
KF848938.1	463	A	G	0.19	KF848938.1	319	T	C	0.21
KF848938.1	499	A	G	0.21	KF848938.1	340	T	C	0.21
KF848938.1	516	A	G	0.2	KF848938.1	356	A	G	0.21
KF848938.1	548	A	G	0.19	KF848938.1	370	A	G	0.22
KF848938.1	591	A	G	0.19	KF848938.1	389	T	C	0.2
KF848938.1	607	A	G	0.18	KF848938.1	409	T	C	0.19
KF848938.1	660	A	G	0.27	KF848938.1	414	T	C	0.3
KF848938.1	670	A	G	0.28	KF848938.1	421	A	G	0.21
KF848938.1	691	A	G	0.23	KF848938.1	463	A	G	0.2
KF848938.1	722	A	G	0.23	KF848938.1	499	A	G	0.19
KF848938.1	744	A	G	0.25	KF848938.1	566	A	G	0.24
KF848938.1	793	A	G	0.17	KF848938.1	597	A	G	0.18
KF848938.1	859	A	G	0.29	KF848938.1	607	A	G	0.2
KF848938.1	898	A	G	0.2	KF848938.1	660	A	G	0.28
KF848938.1	915	T	C	0.21	KF848938.1	670	A	G	0.33
KF848938.1	987	A	G	0.22	KF848938.1	691	A	G	0.23

Roommate	Position	Reference base	Alternative base	Frequency	SRR1705858	Position	Reference base	Alternative base	Frequency
KF848938.1	1031	A	G	0.28	KF848938.1	722	A	G	0.25
KF848938.1	1056	T	C	0.19	KF848938.1	744	A	G	0.22
KF848938.1	1086	A	G	0.21	KF848938.1	759	T	C	0.19
KF848938.1	1100	T	C	0.21	KF848938.1	859	A	G	0.25
KF848938.1	1213	A	G	0.22	KF848938.1	915	T	C	0.27
KF848938.1	1264	T	C	0.21	KF848938.1	987	A	G	0.22
KF848938.1	1280	T	C	0.24	KF848938.1	1031	A	G	0.26
KF848938.1	1358	A	G	0.25	KF848938.1	1043	A	G	0.21
KF848938.1	1366	A	G	0.22	KF848938.1	1056	T	C	0.2
KF848938.1	1398	T	C	0.23	KF848938.1	1086	A	G	0.3
KF848938.1	1421	A	G	0.24	KF848938.1	1089	A	G	0.22
KF848938.1	1460	A	G	0.37	KF848938.1	1105	A	G	0.22
KF848938.1	1482	A	G	0.25	KF848938.1	1209	A	G	0.27
KF848938.1	1517	A	G	0.24	KF848938.1	1213	A	G	0.24
KF848938.1	1520	T	C	0.27	KF848938.1	1264	T	C	0.27
KF848938.1	1600	T	C	0.35	KF848938.1	1280	T	C	0.25
KF848938.1	1604	T	C	0.31	KF848938.1	1281	T	C	0.21
					KF848938.1	1301	A	G	0.22
					KF848938.1	1358	A	G	0.29
					KF848938.1	1366	A	G	0.21
					KF848938.1	1398	T	C	0.23
					KF848938.1	1421	A	G	0.37
					KF848938.1	1460	A	G	0.26
					KF848938.1	1482	A	G	0.23
					KF848938.1	1580	T	C	0.27
					KF848938.1	1604	T	C	0.3

7. Compare the control results to your roommate's results

We obtained the following results.

	mean	**SD**	**mean + 3SD**	**mean - 3SD**
SRR1705858	0,256491	0,071726	0,471669	0,041313
SRR1705859	0,236923	0,052376	0,394051	0,079795
SRR1705860	0,250328	0,078038	0,016215	0,484441

Using these "**mean + 3SD"** let's see the super rare SNPs in our sample:

- POS: 307, C → T, frequency: 0.94
- POS: 1458, T → C, frequency: 0.84

In case of roommate sample using common (high-frequency, >90%) variants for this calculation is not appropriate because of lost of important data.

8. Epitope mapping

We are working with partial cds in our project. The mutation C → T (Pro>Ser) corresponding to 102 aminoacidic residual which corresponds to the epitope D (Munoz et al).

The other mutations are in the inter-epitopes parts.

You can see all the mutations linted in the table below:

Roommate	KF848938.1	KF848938.1	KF848938.1	KF848938.1	KF848938.1	KF848938.1	KF848938.1
Nucleic position	72	117	307	774	999	1260	1458
Reference base	A	C	C	T	C	A	T
Alternative base	G	T	T	C	T	C	C
Amino acid change	Thr (ACA → ACG)	Ala (GCC → GCT)	Pro → Ser (CCG → TCG)	Phe (TTT → TTC)	Gly (GGC → GGT)	Leu (CTA → CTC)	Tyr (TAT → TAC)
Synonym or not	syn	syn	non/missence	syn	syn	syn	syn
Variant frequency (%)	99.96	99.82	0.94	99.96	99.86	99.94	0.84
Amino acid position	24	39	102	258	333	420	486
Affect the epitope regions	-	-	+/ D epitope	-	-	-	-

9. Variant Calling Workflow. Snakefile

Description

This Snakefile represents a workflow for variant calling in a genome using sequencing data. The process includes downloading raw data, indexing the reference genome, aligning reads, converting to BAM format, sorting, indexing the sorted BAM, and finally calling variants using VarScan.

Usage

1. **Download Raw Data (Fastq)**

- Raw data is downloaded from the NCBI SRA repository and stored in the `raw_data` folder.
- You can change the SRA identifier (e.g., SRR1705851) in the `rule download_data` block.

2. **Index Reference Genome (BWA)**

- The reference genome is indexed using BWA and saved in the `alignments` folder.
- The reference genome should be provided in fasta format and placed in the `alignments` folder.

3**. Align Reads (BWA mem)**

Reads are aligned to the reference genome and saved in SAM format in the alignments folder.
Input data for this step includes the reference genome and raw data.

4. Convert to BAM Format (samtools)

The SAM file is converted to BAM format and saved in the alignments folder.

5. **Sort BAM (samtools)**

The BAM file is sorted and saved in the alignments folder.

6. **Index Sorted BAM (samtools)**

The sorted BAM is indexed for quick access and saved in the alignments folder.

7. **Call Variants (VarScan)**

Using VarScan, a VCF file is created containing information about the called variants and saved in the variant_calling folder.

Requirements

Tool requirements:

- BWA
- samtools
- VarScan
- Java Runtime Environment (JRE)

In the working directory where you execute Snakemake, make sure to include all the files necessary for your pipeline to run. In your case, this includes:

- **Snakefile**: Your Snakemake rules file (Snakefile).
- **influenza_hemagglutinin_gene.fasta**: File containing the reference sequence of the hemagglutinin gene in FASTA format.

- **SRR1705851.fastq.gz**: File with the raw sequencing data (assumed to be downloaded by your pipeline).
- **VarScan.v2.4.0.jar**: Jar file for VarScan.

Before running the pipeline, ensure that these files are present in the working directory.

Execution

```
conda init bash
source ~/.bashrc
mkdir Project2/ # if you don't have Project2 folder
cd Project2/
# download all required files in Project2 folder
ls Project2/
# Snakefile
# influenza_hemagglutinin_gene.fasta
# SRR1705851.fastq.gz
# VarScan.v2.4.0.jar
snakemake --cores 1 -p SRR1705851_VarScan_results.vcf # for snakemake version above 5.10 it is needed to specify cores
```

```
# output files
SRR1705851.bam
SRR1705851.fastq.gz
SRR1705851.sam
Snakefile
VarScan.v2.4.0.jar
VarScan_results.vcf
alignment_SRR1705851_sorted.bam
influenza_hemagglutinin_gene.fasta
influenza_hemagglutinin_gene.fasta.amb
influenza_hemagglutinin_gene.fasta.ann
influenza_hemagglutinin_gene.fasta.bwt
influenza_hemagglutinin_gene.fasta.fai
influenza_hemagglutinin_gene.fasta.pac
influenza_hemagglutinin_gene.fasta.sa
my.mpileup
```

Snakefile code

```
#Data downloaded from: http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/001/SRR1705851/SRR1705851.fastq.gz

# Default rule to run entire workflow, only works once inputs/outputs correctly filled in all rules
rule all:
    input: "SRR1705851_VarScan_results.vcf"

# Download raw sequence data
rule download_data:
    output: "SRR1705851.fastq.gz"
    shell:
        "wget http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/001/SRR1705851/SRR1705851.fastq.gz -O {output}"

# Create an index for the reference genome for bwa
rule index_genome_bwa:
    input: "influenza_hemagglutinin_gene.fasta"
    output:
        expand("influenza_hemagglutinin_gene.fasta.{ext}", ext=['sa', 'amb', 'ann', 'pac', 'bwt'])
    shell:
        "bwa index {input}"

# Map the raw reads to the reference genome
rule map_reads:
    input:
        genome = "influenza_hemagglutinin_gene.fasta",
        reads = "SRR1705851.fastq.gz",
        idxfile = expand("influenza_hemagglutinin_gene.fasta.{ext}", ext=['sa', 'amb', 'ann', 'pac', 'bwt'])
    output: "SRR1705851.sam"
    shell:
        "bwa mem -t 4 {input.genome} {input.reads} > {output}"

# Convert .sam to .bam file
rule samtools_import:
    input:
        samfile="SRR1705851.sam"
    output:"SRR1705851.bam"
    shell:
        """
        samtools view -S -b {input.samfile} -o {output}
        """
        # original command with samtools v1.9
        ## samtools import {input.index} {input.samfile} {output}
        # but it gave segmentation fault error with samtools v1.10, so here we use samtools view instead

# Sort the bam alignment file
rule samtools_sort:
    input: "SRR1705851.bam"
    output: "alignment_SRR1705851_sorted.bam"
    shell:
        "samtools sort {input} -o {output}"

# Create an index for the sorted bam file
# again, this is different from the above indexing steps
rule bwa_index_sorted:
    input: "alignment_SRR1705851_sorted.bam"
    output: "alignment_SRR1705851_sorted.bam.bai"
    shell: "samtools index {input}"

# Generate pileup file with samtools, then call variants with bcftools
# From samtools doc: 'Pileup format consists of TAB-separated lines, with each line representing the pileup of reads at a single genomic position.'
rule samtools_mpileup:
    input:
        index="influenza_hemagglutinin_gene.fasta",
        sorted="alignment_SRR1705851_sorted.bam",
    output:
        "SRR1705851_VarScan_results.vcf"
    shell:
        """
        samtools mpileup -f {input.index} {input.sorted} -d 0 > my.mpileup && \
        java -jar VarScan.v2.4.0.jar mpileup2snp my.mpileup --min-var-freq 0.95 --variants --output-vcf 1 > {output}
        """
```