Supplementary materials

1. Inspect the data from your roommate

1) Download raw reads using sratoolkit.2.10.8-ubuntu64

```
fasterq-dump -O . SRR1705851
```

```
Output message:
spots read       : 358,265
reads read       : 358,265
reads written    : 358,265
```

2) Manual inspection of raw reads:

2.1) calculate number of reads

```
python3 -c "print($(cat SRR1705851.fastq | wc -l)/4)"
358265
```

2.2) Read length varies from 35 to 151

```
 for line in $(sed -n '2~4p' SRR1705851.fastq); do echo "${#line}"; done |
sort | uniq | less
```

3) Check quality of reads

```
fastqc -o .  SRR1705851.fastq
```

4) Create a fasta file containing sequence for the influenza hemagglutinin gene (https://www.ncbi.nlm.nih.gov/nuccore/KF848938.1?report=fasta).

```
 nano KF848938_ref.fasta
```

2. Aligning reads to the reference gene

1) Index reference gene:

```
bwa index KF848938_ref.fasta -p ref
```

2) Align reads to the indexed gene and convert resulting alignment into bam format, sort it

```
bwa mem ref ./SRR1705851.fastq | samtools view -S -b - | samtools sort -
-o ./SRR1705851.sorted.bam
```

3) Check if the reference was correct - estimate fraction of mapped/unmapped reads

From the command `samtools view`, the option -f4 reports alignments with 4 bits in the FLAG field which specifies cases of unmapped reads, while option -F4 yields aligned reads.

3.1) Get number of unmapped reads:

```
samtools view -f4 SRR1705851.sorted.bam | wc -l
233
```

233 reads were unmapped.

3.2) Calculate amount of aligned reads:

```
samtools view -F4 SRR1705851.sorted.bam | wc -l
361116
```

3.3) Estimate percentage of unmapped reads:

```
python3 -c "print(round($(samtools view -f4 SRR1705851.sorted.bam | wc
-l)/$(samtools view -F4 SRR1705851.sorted.bam | wc -l)*100,4))"
0.0645
```

About 0.06 % of reads were not mapped to the reference sequence. This value corresponds to the fraction of mapped reads (99.94%, 361116 reads) from basics statistics obtained by `samtools flagstat`

3.4) Compare with basics statistics :

```
samtools flagstat SRR1705851.sorted.bam
361349 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
3084 + 0 supplementary
0 + 0 duplicates
361116 + 0 mapped (99.94% : 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. Look for common variants with VarScan

1) Estimate coverage

Before running samtools mpileup, we should estimate the coverage of sequencing and then specify appropriate value in -d option (max_depth) while creating a mpileup file to catch rare variants in experimental samples during SNP calling.

Coverage = Number_Of_Reads * Average_Read_Length/Ref_Length

From samtools manual = The output is TAB-delimited with each line consisting of reference sequence name, sequence length, # mapped read-segments and # unmapped read-segments. It is written to stdout. Note this may count reads multiple times if they are mapped more than once or in multiple fragments.

```
samtools idxstats SRR1705851.sorted.bam
KF848938.1	1665	361116	0
*	0	0	233
```

1.1) get average read length:

```
samtools stats SRR1705851.sorted.bam | grep ^SN | cut -f 2- | grep -e
'average length'
average length: 147
```

1.2) get length of reference sequence:

```
samtools idxstats SRR1705851.sorted.bam | cut -f2 | head -1
1665
```

or:

```
samtools coverage SRR1705851.sorted.bam | cut -f3
endpos
1665
```

1.3) Get number of mapped read-segments:

```
samtools idxstats SRR1705851.sorted.bam | cut -f3 | head -1
361116
```

or

```
samtools coverage SRR1705851.sorted.bam | cut -f4
numreads
361116
```

1.4) calculate coverage using the following formula:

Coverage = Number_Of_Reads * Average_Read_Length/Ref_Length = 361116 *147 / 1165 = 45565

1.5) Another way to calculate coverage:

Numebr_of_Reads = Number_of_Reads_before_mapping - Number_of_umapped_Reads = 358265 - 233 = 358032

Coverage = Number_Of_Reads * Average_Read_Length/Ref_Length = 358032*147/1165 = 45176

As a number of mapped reads include supplementary reads, we decided to use the last value of coverage (see 1.5)).

1.5) Compare with maximum coverage obtained using bedtools:

```
genomeCoverageBed -ibam SRR1705851.sorted.bam -d | cut -f3 | sort -rn |
head -1
44522
```

or with mean coverage:

```
samtools coverage SRR1705851.sorted.bam | cut -f7
meandepth
31212.7
```

2) Indexing bam file and running samtools mpileup with -d option 50000

```
samtools index ./SRR1705851.sorted.bam
samtools mpileup -f KF848938_ref.fasta -d 50000 SRR1705851.sorted.bam >
./SRR1705851mpileup50
```

3) SNP calling using VarScan with minimum variant frequency of 95%:

```
java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
mpileup2snp SRR1705851mpileup50 --min-var-freq 0.95 --variants
--output-vcf 1 > SRR1705851_50_95.vcf
```

```
Min coverage:   8
Min reads2:     2
Min var freq:   0.95
Min avg qual:   15
P-value thresh: 0.01
Reading input from SRR1705851mpileup50
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)
```

5 variants are reported.

Visualization of variants in the IGV browser showed that all variants occurring with frequency > 95% lead to synonymous mutations and therefore could not impair the antigen recognition by antibodies derived from the vaccine.

4. Look for rare variants with VarScan

1) SNP calling using VarScan with minimum variant frequency of 0.1 % to identify rare variants:

```
java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
mpileup2snp SRR1705851mpileup50 --min-var-freq 0.001 --variants
--output-vcf 1 > SRR1705851_50_001.vcf

Min coverage:   8
Min reads2:     2
Min var freq:   0.001
Min avg qual:   15
P-value thresh: 0.01
Reading input from SRR1705851mpileup50
1665 bases in pileup file
23 variant positions (21 SNP, 2 indel)
0 were failed by the strand-filter
21 variant positions reported (21 SNP, 0 indel)
```

21 SNPs were reported with frequency varying from 0.17% to 99.96% (**Table S1**):

2) Get frequency

```
grep -v '#' SRR1705851_50_001.vcf | awk -F':' '{print $20}' |  sort -n|
uniq
0.17%
0.18%
0.19%
0.2%
0.21%
0.22%
0.23%
0.84%
0.94%
99.82%
99.86%
99.94%
99.96%
```

5. Inspect and align the control sample sequencing data

1) Downloading control sample sequencing data and unzipping 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
```

```
gunzip SRR1705858.fastq.gz SRR1705859.fastq.gz SRR1705860.fastq.gz
```

2) Checking the quality of reads using FastQC:

```
fastqc -o .  SRR1705858.fastq SRR1705859.fastq SRR1705860.fastq
```

3) Estimate the coverage in the control samples:

3.1) Calculate average read length in fastq files:

```
awk '{if(NR%4==2) {count++; bases += length} } END{print bases/count}'
SRR1705859.fastq
148.446
```

```
awk '{if(NR%4==2) {count++; bases += length} } END{print bases/count}'
SRR1705858.fastq
148.561
```

```
awk '{if(NR%4==2) {count++; bases += length} } END{print bases/count}'
SRR1705860.fastq
148.703
```

3.2) Calculate number of reads in each file:

```
python3 -c "print($(cat SRR1705858.fastq | wc -l)/4)"
256586.0
```

```
python3 -c "print($(cat SRR1705859.fastq | wc -l)/4)"
233327.0
```

```
python3 -c "print($(cat SRR1705860.fastq | wc -l)/4)"
249964.0
```

3.3) Average coverage (rough estimation):

for SRR1705858: 148.561*256586/1665 = 22894
for SRR1705859: 148.446 *233327/ 1665 = 20802
for SRR1705860: 148.703 * 249964/1665 = 22324

4) Aligning, compressing ro binary format and sorting:

```
bwa mem ref ./SRR1705858.fastq | samtools view -S -b - | samtools sort -
-o ./SRR1705858.sorted.bam

bwa mem ref ./SRR1705859.fastq | samtools view -S -b - | samtools sort -
-o ./SRR1705859.sorted.bam

bwa mem ref ./SRR1705860.fastq | samtools view -S -b - | samtools sort -
-o ./SRR1705860.sorted.bam
```

5) Indexing sorted alignment files

```
samtools index ./SRR1705858.sorted.bam
samtools index ./SRR1705859.sorted.bam
samtools index ./SRR1705860.sorted.bam
```

6) Coverage estimation after mapping:

6.1) Get number of unmapped reads:

```
samtools view -f4 SRR1705858.sorted.bam | wc -l
86
samtools view -f4 SRR1705859.sorted.bam | wc -l
76
samtools view -f4 SRR1705860.sorted.bam | wc -l
76
```

6.2) Get average reads length:

```
samtools stats SRR1705858.sorted.bam | grep ^SN | cut -f 2- | grep -e 'average length'
average length: 148

samtools stats SRR1705859.sorted.bam | grep ^SN | cut -f 2- | grep -e 'average length'
average length: 148

samtools stats SRR1705860.sorted.bam | grep ^SN | cut -f 2- | grep -e 'average length'
average length: 148
```

6.3) Calculating the coverage:

Numebr_of_Reads = Number_of_Reads_before_mapping - Number_of_umapped_Reads
Coverage = Number_Of_Reads * Average_Read_Length/Ref_Length

for SRR1705858: (256586 - 86)*148/1165 = 32585
for SRR1705859: (233327 - 76) *148/1165 = 29631
for SRR1705860: (249964 - 76)*148/1165 = 31745

6. Searching for rare variants in the reference files using VarScan.

1) Creating mpileup files specifying max depth as 30000:

```
 samtools mpileup -f KF848938_ref.fasta -d 35000 SRR1705858.sorted.bam >
./SRR1705858mpileup35

samtools mpileup -f KF848938_ref.fasta -d 35000 SRR1705859.sorted.bam >
./SRR1705859mpileup35

samtools mpileup -f KF848938_ref.fasta -d 35000 SRR1705860.sorted.bam >
./SRR1705860mpileup35
```

2) Running VarScan with minimum variant frequency of 0.001:

2.1) SRR1705858 sample:

```
java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
mpileup2snp SRR1705858mpileup35 --min-var-freq 0.001 --variants
--output-vcf 1 > SRR1705858_35_001.vcf

Min coverage:   8
Min reads2:     2
Min var freq:   0.001
Min avg qual:   15
P-value thresh: 0.01
Reading input from SRR1705858mpileup35
1665 bases in pileup file
58 variant positions (58 SNP, 0 indel)
1 were failed by the strand-filter
57 variant positions reported (57 SNP, 0 indel)
```

2.2) SRR1705859 sample:

```
java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
mpileup2snp SRR1705859mpileup35 --min-var-freq 0.001 --variants
--output-vcf 1 > SRR1705859_35_001.vcf

Min coverage:   8
Min reads2:     2
Min var freq:   0.001
Min avg qual:   15
P-value thresh: 0.01
```

```
Reading input from SRR1705859mpileup35
1665 bases in pileup file
54 variant positions (54 SNP, 0 indel)
2 were failed by the strand-filter
52 variant positions reported (52 SNP, 0 indel)
```

2.3) SRR1705860 sample:

```
java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
mpileup2snp SRR1705860mpileup35 --min-var-freq 0.001 --variants
--output-vcf 1 > SRR1705860_35_001.vcf
```

```
Min coverage:   8
Min reads2:     2
Min var freq:   0.001
Min avg qual:   15
P-value thresh: 0.01
Reading input from SRR1705860mpileup35
1665 bases in pileup file
61 variant positions (61 SNP, 0 indel)
0 were failed by the strand-filter
61 variant positions reported (61 SNP, 0 indel)
```

3) Create Google Sheets containing the reference base, position, alternative base, and frequency.
https://docs.google.com/spreadsheets/d/11C0bkxm5MsZRW80NHc1LA9WgdzjyNA-5jG3BIquhLps/edit?usp=sharing

All identified variants in the control samples are listed in the **Tabe S2 - S4**

3.1) Get frequency

```
grep -v '#' SRR1705858_35_001.vcf | awk -F':' '{print $20}'
grep -v '#' SRR1705859_35_001.vcf | awk -F':' '{print $20}'
grep -v '#' SRR1705860_35_001.vcf | awk -F':' '{print $20}'
```

3.2) Get the reference position, base and alternative base

```
grep -v '#' SRR1705858_35_001.vcf | awk -F'\t' '{print  $2,$4,$5}'
grep -v '#' SRR1705859_35_001.vcf | awk -F'\t' '{print  $2,$4,$5}'
grep -v '#' SRR1705860_35_001.vcf | awk -F'\t' '{print  $2,$4,$5}'
```

4) Create files for subsequent calculations containing the following information from vcf file: POS, REF, ALT and FREQ

4.1) SRR1705860

```
grep -v '#' SRR1705860_35_001.vcf | awk -F'\t' '{print  $2,$4,$5}' >
SRR1705860_35_001.txt_1

grep -v '#' SRR1705860_35_001.vcf | awk -F':' '{print $20}' | cut -f1 -d
'%' > SRR1705860_35_001.txt_2

paste -d ' ' SRR1705860_35_001.txt_1 SRR1705860_35_001.txt_2 >
SRR1705860_35_001.txt
```

4.2) SRR1705859

```
grep -v '#' SRR1705859_35_001.vcf | awk -F'\t' '{print  $2,$4,$5}' >
SRR1705859_35_001.txt_1

grep -v '#' SRR1705859_35_001.vcf | awk -F':' '{print $20}' | cut -f1 -d
'%' > SRR1705859_35_001.txt_2

paste -d ' ' SRR1705859_35_001.txt_1 SRR1705859_35_001.txt_2 >
SRR1705859_35_001.txt
```

4.3) SRR1705858

```
grep -v '#' SRR1705858_35_001.vcf | awk -F'\t' '{print  $2,$4,$5}' >
SRR1705858_35_001.txt_1

grep -v '#' SRR1705858_35_001.vcf | awk -F':' '{print $20}' | cut -f1 -d
'%' > SRR1705858_35_001.txt_2

paste -d ' ' SRR1705858_35_001.txt_1 SRR1705858_35_001.txt_2 >
SRR1705858_35_001.txt
```

7. Compare the control results to experimental results

1) Calculate statistics of variant frequencies from control samples using python 3:

```
import pandas as pd
import numpy as np

path = '/home/rybina/BI2020prak/Project2/'
df_SRR1705858 = pd.read_csv(path+'SRR1705858_35_001.txt', sep = ' ', header = None)
```

```
df_SRR1705859 = pd.read_csv(path+'SRR1705859_35_001.txt', sep = ' ', header = None)
df_SRR1705860 = pd.read_csv(path+'SRR1705860_35_001.txt', sep = ' ', header = None)

data_summary = {
             'Mean_Frequency':[np.mean(df_SRR1705858[df_SRR1705858[3] < 0.90][3]),
                              np.mean(df_SRR1705859[df_SRR1705859[3] < 0.90][3]),
                              np.mean(df_SRR1705860[df_SRR1705860[3] < 0.90][3]),
                              (np.mean(df_SRR1705860[df_SRR1705860[3] < 0.90][3])+
                              np.mean(df_SRR1705859[df_SRR1705859[3] < 0.90][3])+
                              np.mean(df_SRR1705858[df_SRR1705858[3] < 0.90][3]))/3],
             'Std_Frequency':[np.std(df_SRR1705858[df_SRR1705858[3] < 0.90][3]),
                             np.std(df_SRR1705859[df_SRR1705859[3] < 0.90][3]),
                             np.std(df_SRR1705860[df_SRR1705860[3] < 0.90][3]),
                              (np.std(df_SRR1705860[df_SRR1705860[3] < 0.90][3])+
                              np.std(df_SRR1705859[df_SRR1705859[3] < 0.90][3])+
                              np.std(df_SRR1705858[df_SRR1705858[3] < 0.90][3]))/3]}

df_summary= pd.DataFrame(data_summary, index =
['SRR1705858','SRR1705859','SRR1705860','Average'])
df_summary
```

```
       Mean_Frequency Std_Frequency
SRR1705858     0.257544       0.071015
SRR1705859     0.237308       0.051966
SRR1705860     0.250820       0.078329
Average 0.248557       0.067103
```

2) Find SNPs in experimental sample that are likely to be not sequencing errors:

2.1) Prepare file containing the reference base, position, alternative base, and frequency for experimental sample:

```
grep -v '#' SRR1705851_50_001.vcf | awk -F'\t' '{print  $2,$4,$5}' >
SRR1705851_50_001.txt_1

grep -v '#' SRR1705851_50_001.vcf | awk -F':' '{print $20}' | cut -f1 -d '%' >
SRR1705851_50_001.txt_2

paste -d ' ' SRR1705851_50_001.txt_1 SRR1705851_50_001.txt_2 >
SRR1705851_50_001.txt
```

2.2.1) Using python3, find SNPs in experimental sample which frequency are more than 3 standard deviations away from averages of reference data:

```
df_SRR1705851 = pd.read_csv(path+'SRR1705851_50_001.txt', sep = ' ', header = None)

condition = (np.array(df_SRR1705851[3]) < np.array((df_summary[-1:]['Mean_Frequency'] +
3*df_summary[-1:]['Std_Frequency']))) \
& (np.array(df_SRR1705851[3]) > np.array((df_summary[-1:]['Mean_Frequency'] -
3*df_summary[-1:]['Std_Frequency'])))
```

```
df_SRR1705851_filtered = df_SRR1705851[~condition]
df_SRR1705851_filtered = df_SRR1705851_filtered.rename(columns =
{0:'POS',1:'REF',2:'ALT',3:'FREQ'})
```

2.2.2) Using R, find SNPs in experimental sample which frequency are more than 3 standard deviations away from averages of reference data:

#read data (shown only for sample)

```
roommate <-
read.table("/home/daria/Documents/IB_fall2020/Project_2/roommate.tsv",
header = F, sep = "\t")
```

#select only rare variants

```
rare_roommate <- roommate[roommate$V5 < 1, ]
```

#calculate mean and sd in replicates (shown only for one of controls) and find interval

```
mean_sd_control58 <- control58 %>%
  summarise(mean_value = mean(V5),
            std = sd(V5))

threshold_58 <- mean_sd_control58$std * 3
```

#find variants that lie more than 3 sd away from replicate (shown only for one of controls)

```
rare_roommate_58 <- rare_roommate[abs(rare_roommate$V5 -
mean_sd_control58$mean_value)  > threshold_58, ]
```

#two SNPs were identified (same for each control):

```
KF848938.1  307  C  T 0.94
KF848938.1 1458  T  C 0.84
```

3) Resulting data frame df_SRR1705851_filtered was updated by column MUTATION after exploring SNPs in IGV browser (**Figure S1- S7**):

POS	REF	ALT	FREQ (%)	MUTATION
72	A	G	99.96	T24T
117	C	T	99.82	A34A
307	C	T	0.94	P103S
774	T	C	99.96	F258F
999	C	T	99.86	G333G
1260	A	C	99.94	L420L
1458	T	C	0.84	T485T

There are 2 rare SNPs which frequencies are more than 3 standard deviations away from the averages of the control samples. Another 5 mutations of high frequency are common.

4) Finding common mutations in control samples using python and VarScan (command “compare”)

4.1) Using VarScan.v2.4.1.jar compare command

```
 java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
compare SRR1705859_35_001.vcf SRR1705858_35_001.vcf intersect
SRR1705859_SRR1705858.intersect.vcf

Picked up _JAVA_OPTIONS: -Xmx500g
File 1: SRR1705859_35_001.vcf
File 2: SRR1705858_35_001.vcf
Loading positions from file 1
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
Loading positions from file 2
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
Done
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
75 total positions
18 positions unique to file 1
23 positions unique to file 2
34 positions shared

java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
compare SRR1705860_35_001.vcf SRR1705859_SRR1705858.intersect.vcf
intersect control.intersect.vcf

Picked up _JAVA_OPTIONS: -Xmx500g
File 1: SRR1705860_35_001.vcf
File 2: SRR1705859_SRR1705858.intersect.vcf
Loading positions from file 1
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
Loading positions from file 2
Done
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
62 total positions
28 positions unique to file 1
1 positions unique to file 2
33 positions shared
```

```
 java -jar /home/rybina/BI2020prak/Project1/varscan/VarScan.v2.4.1.jar
compare SRR1705851_50_001.vcf control.intersect.vcf intersect
total.intersect.vcf
```

```
Picked up _JAVA_OPTIONS: -Xmx500g
File 1: SRR1705851_50_001.vcf
File 2: control.intersect.vcf
Loading positions from file 1
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
Loading positions from file 2
Done
Warning: Unable to parse chrom/position from #CHROM     POS     ID
REF     ALT     QUAL    FILTER  INFO    FORMAT  Sample1
43 total positions
10 positions unique to file 1
22 positions unique to file 2
11 positions shared
```

Overall, 11 variants were shared between 3 control and 1 experimental sample.

4.2) Using python 3:

```
df_SRR1705860[[0,1,2]].merge(df_SRR1705859[[0,1,2]].merge(df_SRR1705858[[0,1,2]].merge(df_SRR1705851[[0,1,2]])))

# Find Freq of common mutations in experimental sample
df_SRR1705851[df_SRR1705851[0].isin(list(df_SRR1705860[[0,1,2]].merge(df_SRR1705859[[0,1,2]].merge(df_SRR1705858[[0,1,2]].merge(df_SRR1705851[[0,1,2]])))[0]))].rename(columns = {0:'POS',1:'REF',2:'ALT',3:'FREQ'})
```

POS	REF	ALT	FREQ
254	A	G	0.17
276	A	G	0.17
340	T	C	0.17
691	A	G	0.17
722	A	G	0.20
744	A	G	0.17
859	A	G	0.18
915	T	C	0.19
1086	A	G	0.21
1213	A	G	0.22
1280	T	C	0.18

11 mutations were shared between 3 control and 1 experimental samples (**Table S5**). However frequency of these mutations in experimental sample fell into interval of frequencies probably associated with errors (0.248-3*0.067, 0.248+3*0.067) ~ (0.047, 0.45).

8. Epitope mapping

Revealed mutation P103S is located within the epitope D region (Munoz et al) of hemagglutinin mutations which were not dominant in 2003–2004 in the USA.

9*. Distinguish between PCR and sequencing errors

First way
Taking into account that all three replicates were amplified together, we can assume that they share common errors. Thus, common for three replicates errors are upstream errors, while unique errors occurred during sequencing.

1) Using R we read 3 data frames obtained earlier from VarScan output (fields 1, 2, 4, 5 and frequency):

```
control58 <-
read.table("/home/daria/Documents/IB_fall2020/Project_2/control58.tsv",
header = F, sep = "\t")
control59 <-
read.table("/home/daria/Documents/IB_fall2020/Project_2/control59.tsv",
header = F, sep = "\t")
control60 <-
read.table("/home/daria/Documents/IB_fall2020/Project_2/control60.tsv",
header = F, sep = "\t")
```

2) Find common positions:

```
common_pos_58_59 <- intersect(control58[, c(1:4)], control59[, c(1:4)])
common_pos_controls <- intersect(common_pos_58_59, control60[, c(1:4)])
33 common errors were obtained:
```

```
gene               position    ref sample
1  KF848938.1      165            T        C
2  KF848938.1      183            A        G
3  KF848938.1      216            A        G
4  KF848938.1      218            A        G
5  KF848938.1      222            T        C
6  KF848938.1      254            A        G
7  KF848938.1      276            A        G
8  KF848938.1      340            T        C
9  KF848938.1      356            A        G
10 KF848938.1      370            A        G
11 KF848938.1      409            T        C
12 KF848938.1      414            T        C
13 KF848938.1      421            A        G
14 KF848938.1      463            A        G
```

```
15 KF848938.1     660               A           G
16 KF848938.1     670               A           G
17 KF848938.1     691               A           G
18 KF848938.1     722               A           G
19 KF848938.1     744               A           G
20 KF848938.1     859               A           G
21 KF848938.1     915               T           C
22 KF848938.1     987               A           G
23 KF848938.1     1031              A           G
24 KF848938.1     1056              T           C
25 KF848938.1     1086              A           G
26 KF848938.1     1213              A           G
27 KF848938.1     1264              T           C
28 KF848938.1     1280              T           C
29 KF848938.1     1358              A           G
30 KF848938.1     1398              T           C
31 KF848938.1     1421              A           G
32 KF848938.1     1460              A           G
33 KF848938.1     1482              A           G
```

3) Find mean value and sd of frequencies of common and unique errors for each replicate (shown only for control_58):

```
uniq_58 <- control58 %>%
  filter(!(V2 %in% common_pos$V2)) %>%
  summarise(mean_value = mean(V5),
            std = sd(V5))

common_58 <- control58 %>%
  filter(V2 %in% common_pos$V2) %>%
  summarise(mean_value = mean(V5),
            std = sd(V5))
```

Sample	Mean		Std	
	Common (PCR) errors	Unique (sequencing) errors	Common (PCR) errors	Unique (sequencing) errors
SRR1705858	0.2504545	0.2769231	0.07332484	0.06447023
SRR1705859	0.2366667	0.2373684	0.03829708	0.07186732
SRR1705860	0.2556818	0.2364706	0.07931092	0.07516159
Average	0.247601	0.250254	0.06364428	0.07049971

We can see that mean and sd of upstream and downstream errors only slightly differ, although more precise statistical analysis must be applied.

However the first approach can not distinguish between sequencing errors and errors on late PCR cycles (these errors will not be shared by all sequences).

Second way

Second approach is based on the assumption that sequencing errors may be marked by Illumina as low-quality bases. If that is true, we can check, which positions in alignment are characterized by lower quality in reads than in reference? We can find average quality of reference-supporting bases and average quality of variant-supporting bases in .vcf output after VarScan.

1) Find difference between average quality of reference-supporting bases (V6) and average quality of variant-supporting bases (V7) in R:

```
control58$diff <- control58$V6 - control58$V7
```

2) We assumed that if the difference is bigger than 3 in Phred scale, such error occurred during sequencing:

```
control58 %>%
  filter(diff > 3) %>%
  summarise(mean_value = mean(V5),
            std = sd(V5))
```

3) If the difference is lower than 4 in Phred scale, such error occurred during PCR:

```
control58 %>%
  filter(diff < 4) %>%
  summarise(mean_value = mean(V5),
            std = sd(V5))
```

Sample	Mean		Std	
	diff < 4 (PCR errors)	diff > 3 (sequencing errors)	diff < 4 (PCR errors)	diff > 3 (sequencing errors)
SRR1705858	0.2501923	0.322	0.04746476	0.1934425
SRR1705859	0.23625	0.245	0.05330063	0.0450925
SRR1705860	0.2444681	0.27	0.05512076	0.1296741
Average	0.2436368	0.279	0.05196205	0.1227364

As a result we can see a bigger difference between estimated parameters for PCR and sequencing errors. Actually we are not sure if any of the approaches could differentiate types of errors. But our ideas may be important for further improvement of methods.

10. Tables and Figures

Table S1. All variants with frequency > 0.001 (0.1 %) identified in the clinical sample .

Position	Reference	Alternative	Frequency,%
72	A	G	99.96
117	C	T	99.82
254	A	G	0.17
276	A	G	0.17
307	C	T	0.94
340	T	C	0.17
389	T	C	0.22
691	A	G	0.17
722	A	G	0.2
744	A	G	0.17
774	T	C	99.96
802	A	G	0.23
859	A	G	0.18
915	T	C	0.19
999	C	T	99.86
1043	A	G	0.18
1086	A	G	0.21
1213	A	G	0.22
1260	A	C	99.94
1280	T	C	0.18
1458	T	C	0.84

Table S2. All variants with frequency > 0.001 (0.1 %) identified in the control sample SRR1705858.

Position	Reference	Alternative	Frequency,%
38	T	C	0.66%
54	T	C	0.3%
72	A	G	0.3%
95	A	G	0.24%
117	C	T	0.3%
165	T	C	0.24%
183	A	G	0.3%
216	A	G	0.22%

218	A	G	0.28%
222	T	C	0.26%
235	T	C	0.25%
254	A	G	0.25%
276	A	G	0.22%
297	T	C	0.2%
328	T	C	0.2%
340	T	C	0.23%
356	A	G	0.22%
370	A	G	0.21%
389	T	C	0.23%
409	T	C	0.22%
414	T	C	0.28%
421	A	G	0.18%
426	A	G	0.19%
463	A	G	0.19%
516	A	G	0.2%
566	A	G	0.23%
595	G	T	0.33%
597	A	G	0.19%
660	A	G	0.21%
670	A	G	0.32%
691	A	G	0.21%
722	A	G	0.23%
744	A	G	0.21%
774	T	C	0.31%
859	A	G	0.26%
915	T	C	0.28%
987	A	G	0.22%
1008	T	G	0.27%
1031	A	G	0.28%
1043	A	G	0.24%
1056	T	C	0.2%
1086	A	G	0.33%
1089	A	G	0.22%
1213	A	G	0.24%
1260	A	C	0.3%

1264	T	C	0.26%
1280	T	C	0.25%
1281	T	C	0.22%
1286	T	C	0.2%
1339	T	C	0.41%
1358	A	G	0.26%
1398	T	C	0.2%
1421	A	G	0.31%
1460	A	G	0.34%
1482	A	G	0.24%
1580	T	C	0.25%
1591	T	C	0.29%

Table S3. All variants with frequency > 0.001 (0.1 %) identified in the control sample SRR1705859.

Position	Reference	Alternative	Frequency,%
44	T	C	0.47%
158	A	G	0.24%
165	T	C	0.27%
183	A	G	0.22%
193	A	G	0.22%
216	A	G	0.24%
218	A	G	0.29%
222	T	C	0.25%
254	A	G	0.19%
276	A	G	0.24%
319	T	C	0.23%
340	T	C	0.21%
356	A	G	0.24%
370	A	G	0.21%
398	A	G	0.22%
403	A	G	0.19%
409	T	C	0.19%
414	T	C	0.22%
421	A	G	0.18%
463	A	G	0.19%
499	A	G	0.21%
516	A	G	0.2%

548	A	G	0.19%
591	A	G	0.19%
607	A	G	0.18%
660	A	G	0.26%
670	A	G	0.29%
691	A	G	0.24%
722	A	G	0.22%
744	A	G	0.25%
793	A	G	0.18%
859	A	G	0.3%
898	A	G	0.2%
915	T	C	0.21%
987	A	G	0.22%
1031	A	G	0.28%
1056	T	C	0.19%
1086	A	G	0.21%
1100	T	C	0.21%
1213	A	G	0.22%
1264	T	C	0.21%
1280	T	C	0.24%
1358	A	G	0.25%
1366	A	G	0.22%
1398	T	C	0.23%
1421	A	G	0.24%
1460	A	G	0.37%
1482	A	G	0.25%
1517	A	G	0.24%
1520	T	C	0.27%
1600	T	C	0.35%
1604	T	C	0.31%

Table S4. All variants with frequency > 0.001 (0.1 %) identified in the control sample SRR1705860.

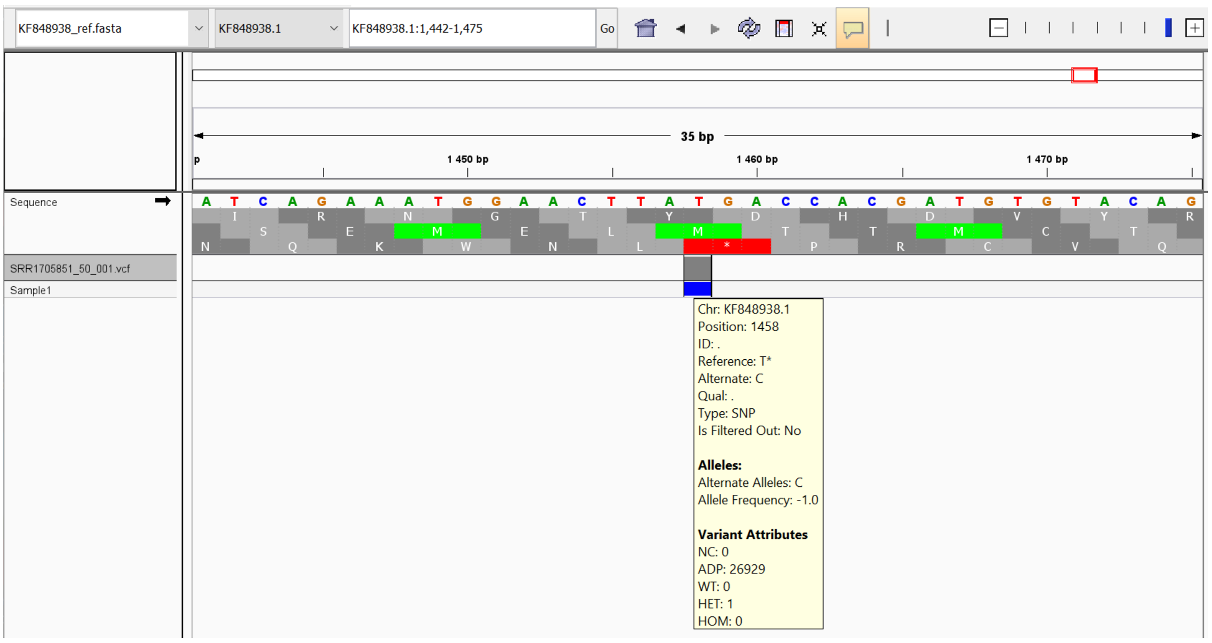
Position	Reference	Alternative	Frequency,%
38	T	C	0.7%
44	T	C	0.5%
95	A	G	0.24%

105	A	G	0.25%
133	A	G	0.22%
158	A	G	0.26%
165	T	C	0.25%
183	A	G	0.23%
199	A	G	0.19%
216	A	G	0.24%
218	A	G	0.23%
222	T	C	0.3%
228	T	C	0.19%
230	A	G	0.19%
235	T	C	0.25%
254	A	G	0.23%
271	A	G	0.21%
276	A	G	0.33%
297	T	C	0.23%
319	T	C	0.21%
340	T	C	0.21%
356	A	G	0.21%
370	A	G	0.22%
389	T	C	0.2%
409	T	C	0.19%
414	T	C	0.3%
421	A	G	0.21%
463	A	G	0.2%
499	A	G	0.19%
566	A	G	0.25%
597	A	G	0.18%
607	A	G	0.19%
660	A	G	0.29%
670	A	G	0.35%
691	A	G	0.25%
722	A	G	0.27%
744	A	G	0.23%
759	T	C	0.19%
859	A	G	0.19%
915	T	C	0.28%

987	A	G	0.22%
1031	A	G	0.26%
1043	A	G	0.21%
1056	T	C	0.2%
1086	A	G	0.3%
1089	A	G	0.22%
1105	A	G	0.22%
1209	A	G	0.27%
1213	A	G	0.24%
1264	T	C	0.27%
1280	T	C	0.25%
1281	T	C	0.21%
1301	A	G	0.22%
1358	A	G	0.29%
1366	A	G	0.21%
1398	T	C	0.23%
1421	A	G	0.37%
1460	A	G	0.26%
1482	A	G	0.23%
1580	T	C	0.27%
1604	T	C	0.3%

Table S5. All shared variants between 3 control and 1 clinical samples

Position	Reference	Alternative	Frequency,%
254	A	G	0.17
276	A	G	0.17
340	T	C	0.17
691	A	G	0.17
722	A	G	0.20
744	A	G	0.17
859	A	G	0.18
915	T	C	0.19
1086	A	G	0.21
1213	A	G	0.22
1280	T	C	0.18

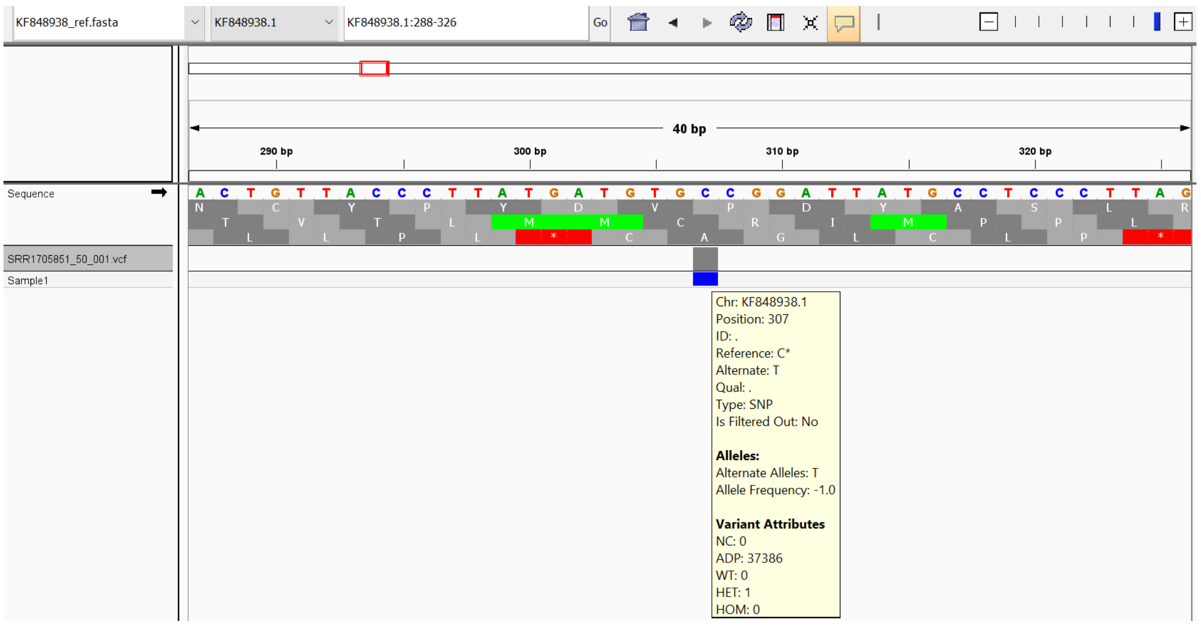

KF848938_ref.fasta | KF848938.1 | KF848938.1:1,442-1,475

Chr: KF848938.1
Position: 1458
ID: .
Reference: T*
Alternate: C
Qual: .
Type: SNP
Is Filtered Out: No

Alleles:
Alternate Alleles: C
Allele Frequency: -1.0

Variant Attributes
NC: 0
ADP: 26929
WT: 0
HET: 1
HOM: 0

Figure S1. Visualization of SNP from roommate’s sequencing data which frequency (0.94 %) is more than 3 standard deviations from the averages in the reference files. The 3rd position in the triplet TAT of coding DNA strand is replaced with C nucleotide resulting in TAC triplet. Both triplets correspond to the amino acid residue Tyrosine (synonymous mutation at position 485).

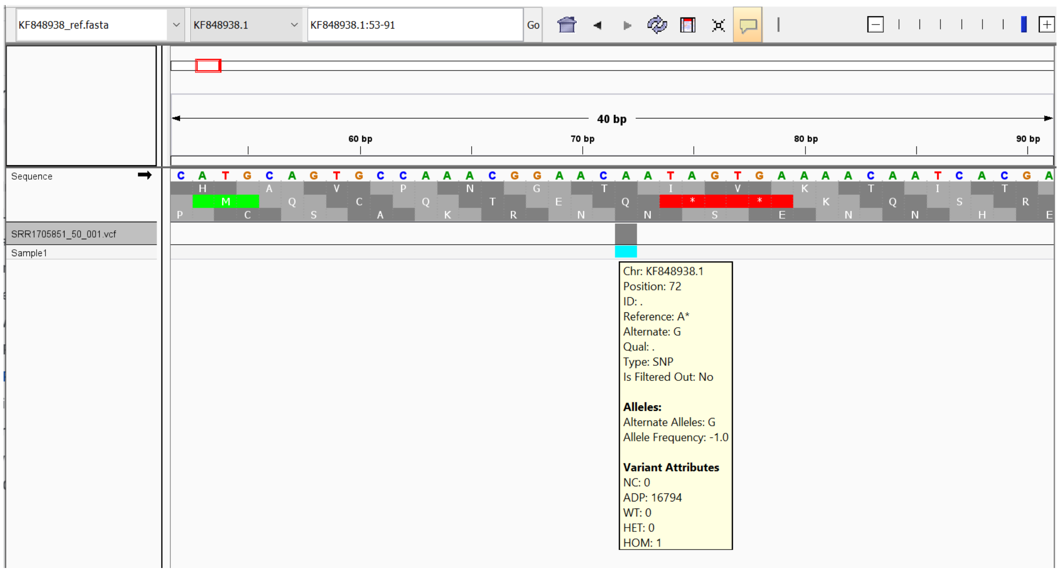

KF848938_ref.fasta | KF848938.1 | KF848938.1:288-326

Chr: KF848938.1
Position: 307
ID: .
Reference: C*
Alternate: T
Qual: .
Type: SNP
Is Filtered Out: No

Alleles:
Alternate Alleles: T
Allele Frequency: -1.0

Variant Attributes
NC: 0
ADP: 37386
WT: 0
HET: 1
HOM: 0

Figure S2. Visualization of SNP from roommate’s sequencing data which frequency (0.84 %) is more than 3 standard deviations from the averages in the reference files. The 3rd position in the triplet CCG of coding DNA strand is replaced with nucleotide T resulting in TCG triplet. Missense mutation P103S: Proline (codon CCG) is replaced with Serine (codon UCG).

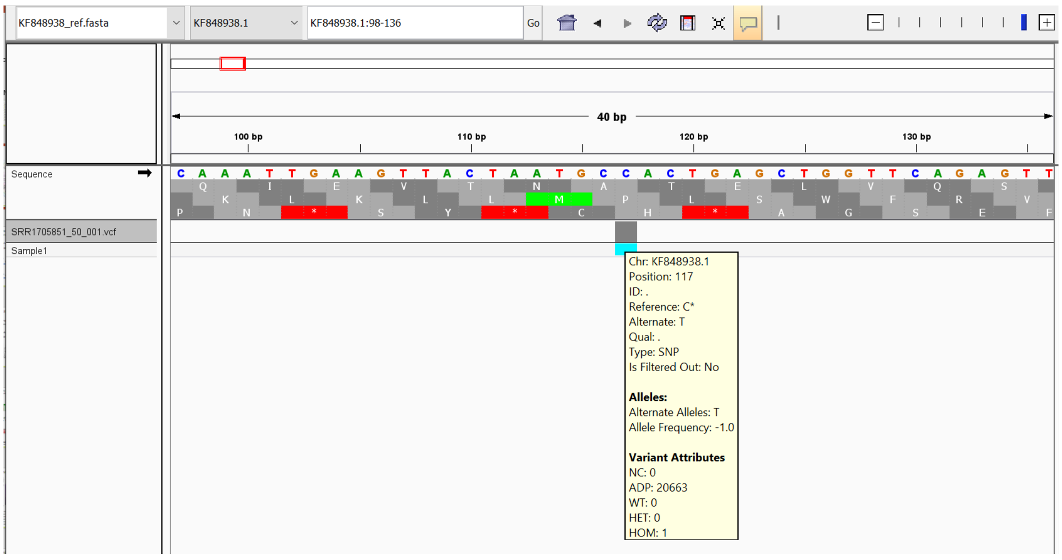


Chr: KF848938.1
Position: 72
ID: .
Reference: A*
Alternate: G
Qual: .
Type: SNP
Is Filtered Out: No

Alleles:
Alternate Alleles: G
Allele Frequency: -1.0

Variant Attributes
NC: 0
ADP: 16794
WT: 0
HET: 0
HOM: 1

Figure S3. Visualization of SNP from roommate's sequencing data which frequency (99.96 %) is more than 3 standard deviations from the averages in the reference files. The 3rd position in the triplet ACA of coding DNA strand is replaced with nucleotide G resulting in ACG triplet. Synonymous mutation at position 24: both codons (ACA and ACG) correspond to the amino acid residue Threonine.

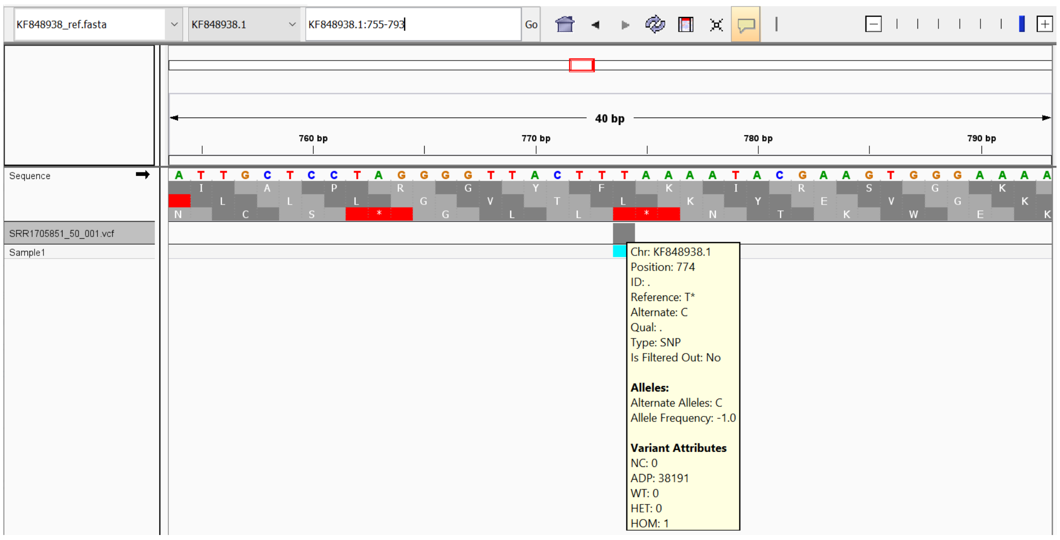


Chr: KF848938.1
Position: 117
ID: .
Reference: C*
Alternate: T
Qual: .
Type: SNP
Is Filtered Out: No

Alleles:
Alternate Alleles: T
Allele Frequency: -1.0

Variant Attributes
NC: 0
ADP: 20663
WT: 0
HET: 0
HOM: 1

Figure S4. Visualization of SNP from roommate's sequencing data which frequency (99.82 %) is more than 3 standard deviations from the averages in the reference files. The 3rd position in the triplet GCC of coding DNA strand is replaced with nucleotide T resulting in GCT triplet. Synonymous mutation at position 39: both codons (GCC and GCU) correspond to the amino acid residue Alanine.

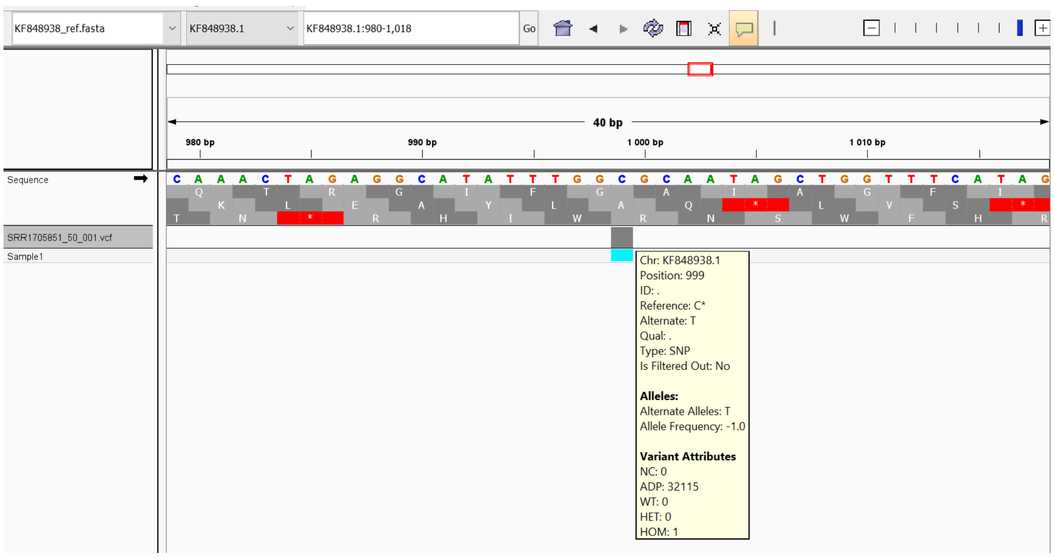


Chr: KF848938.1
Position: 774
ID: .
Reference: T*
Alternate: C
Qual: .
Type: SNP
Is Filtered Out: No

Alleles:
Alternate Alleles: C
Allele Frequency: -1.0

Variant Attributes
NC: 0
ADP: 38191
WT: 0
HET: 0
HOM: 1

Figure S5. Visualization of SNP from roommate's sequencing data which frequency (99.96 %) is more than 3 standard deviations from the averages in the reference files. The 3rd position in the triplet TTT of coding DNA strand is replaced with nucleotide C resulting in TTC triplet. Synonymous mutation at position 258: both codons (UUU and UUC) correspond to the amino acid residue Phenylalanine.

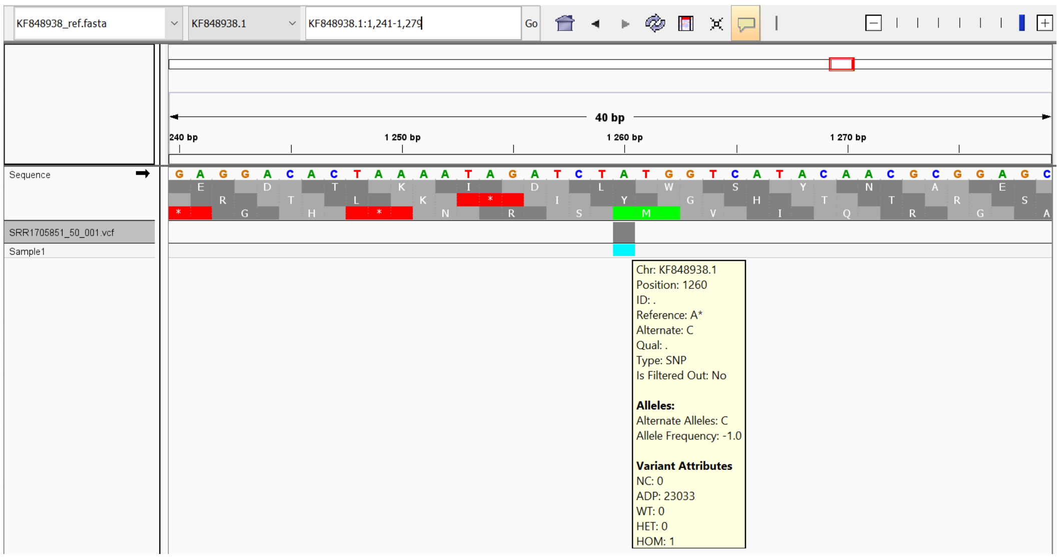


Chr: KF848938.1
Position: 999
ID: .
Reference: C*
Alternate: T
Qual: .
Type: SNP
Is Filtered Out: No

Alleles:
Alternate Alleles: T
Allele Frequency: -1.0

Variant Attributes
NC: 0
ADP: 32115
WT: 0
HET: 0
HOM: 1

Figure S6. Visualization of SNP from roommate's sequencing data which frequency (99.86 %) is more than 3 standard deviations from the averages in the reference files. The 3rd position in the triplet GGC of coding DNA strand is replaced with nucleotide T resulting in GGT triplet. Synonymous mutation at position 333: both codons (GGC and GGU) correspond to the amino acid residue Glycine.



Chr: KF848938.1
Position: 1260
ID: .
Reference: A*
Alternate: C
Qual: .
Type: SNP
Is Filtered Out: No

Alleles:
Alternate Alleles: C
Allele Frequency: -1.0

Variant Attributes
NC: 0
ADP: 23033
WT: 0
HET: 0
HOM: 1

Figure S7. Visualization of SNP from roommate's sequencing data which frequency (99.94 %) is more than 3 standard deviations from the averages in the reference files. The 3rd position in the triplet CTA of coding DNA strand is replaced with nucleotide C resulting in CTC triplet. Synonymous mutation at position 420: both codons (CUA and CUC) correspond to the amino acid residue Leucine.