**What causes antibiotic resistance? Supplementary materials**

1. **Getting the data**

1) Download reference genome sequence

```
wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.fna.gz
```

2) Download annotation

```
wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.gff.gz
```

3) Pair-end reads were downloaded manually

4) Unzip archives in working directory:

```
gunzip GCF_000005845.2_ASM584v2_genomic.fna.gz
gunzip GCF_000005845.2_ASM584v2_genomic.gff.gz
unzip amp_res_1.fastq.zip
unzip amp_res_2.fastq.zip
```

2. **Inspect raw sequencing data manually**

1) Check if a format is correct:

```
head -20 amp_res_1.fastq
```

for .zip archives:

```
zcat amp_res_1.fastq.zip | less | head -20
zcat amp_res_2.fastq.zip | less | head -20
```

2) Count number of reads in each fastq file:
Firstly count number of lines and divide by 4, because each read takes 4 lines in fastq format:

```
python3 -c "print($(zcat amp_res_1.fastq.zip | wc -l)/4)"
```
**Output:** 455876
```
python3 -c "print($(zcat amp_res_2.fastq.zip | wc -l)/4)"
```
**Output:** 455876

3) Find read lengths presented in the data: extract every 4th line starting from the second line that corresponds to each read sequence, calculate length of the string and display unique values:

```
 for line in $(sed -n '2~4p' amp_res_2.fastq); do echo "${#line}"; done | sort | uniq
```
**Output:**101
Average length of read should be 101

**3. Inspect raw sequencing data with fastqc. Filtering the reads.**

1) Installing fastqc:

```
sudo apt install fastqc
```

2) Use fastqc with reads:

```
fastqc -o .  amp_res_1.fastq amp_res_2.fastq
```

Fastqc computed the same value of sequence length (101) and number of reads (455876) as we obtained in the previous step using bash commands.

For both fastq files, FastQC output reported  warnings in *per base sequence content* and *per sequence GC content*.

Test for *per base sequence quality* failed for both samples (Fig. S1). This result may be due to the fact that the quality of calls on most platforms will degrade as the run progresses.

*Per tile sequence quality* was reported as a warning for amp_res_2.fastq and was failed in case of amp_res_1.fastq. Maybe it is due to some transient problem with the run (bubbles passing through a flowcell for example). We can see at the per-tile quality plot that there are indeed some red areas that can be caused by a short decline of quality during sequencing (Fig. S2). In this case trimming is not an excellent solution, as far as we can trim a great part of reads, which spoils further alignment. One possible solution is to mask the bases with bad quality during alignment and to be more accurate during sequencing next time!

**4. (Optional, 1 bonus point) Filtering the reads**

1) Install trimmomatic:

```
sudo apt install trimmomatic
```

2) Trimming with quality score of 20:

```
java -jar ~/Trimmomatic-0.39/trimmomatic-0.39.jar PE -phred33
amp_res_1.fastq amp_res_2.fastq amp_res_1.trimmed.fastq
amp_res_1.trimmed.unpaired.fastq amp_res_2.trimmed.fastq
amp_res_2.trimmed.unpaired.fastq LEADING:20 TRAILING:20
SLIDINGWINDOW:10:20 MINLEN:20
```

**Output message:** Input Read Pairs: 455876 Both Surviving: 446259 (97.89%) Forward Only Surviving: 9216 (2.02%) Reverse Only Surviving: 273 (0.06%) Dropped: 128 (0.03%) TrimmomaticPE: Completed successfully

3) Calculate number of reads and their length in output files *.trimmed.fastq combining bash and python 3:

```
python3 -c "print($(cat amp_res_1.trimmed.fastq | wc -l)/4)"
```
**Output:** 446259.0
```
python3 -c "print($(cat amp_res_2.trimmed.fastq | wc -l)/4)"
```
**Output:** 446259.0

Read length varies from 20 to 101 in both *.trimmed.fastq files:
```
for line in $(sed -n '2~4p' amp_res_1.trimmed.fastq); do echo "${#line}";
done | sort -n | uniq
for line in $(sed -n '2~4p' amp_res_2.trimmed.fastq); do echo "${#line}";
done | sort -n | uniq
```

4) Run fastqc on modified fastq files to check how trimming changed the quality of the data:
```
fastqc -o . amp_res_1.trimmed.fastq amp_res_2.trimmed.fastq
```

*Per base sequence quality* passed the test (warning has disappeared): mean quality score is higher than 28 (phred33 encoding) in both files (Fig. S3). Sequence length distribution is now skewed, sequence length varies from 20 to 101 (101 is the mode of sequence length distribution). According to FastQC report, trimming has improved the overall quality of the read sequence data.

5) Trimming with quality score of 30:
```
java -jar ~/Trimmomatic-0.39/trimmomatic-0.39.jar PE -phred33
amp_res_1.fastq amp_res_2.fastq amp_res_1.trimmed30.fastq
amp_res_1.trimmed30.unpaired.fastq amp_res_2.trimmed30.fastq
amp_res_2.trimmed30.unpaired.fastq LEADING:30 TRAILING:30
SLIDINGWINDOW:10:30 MINLEN:20
```

**Output message:** Input Read Pairs: 455876 Both Surviving: 376340 (82.55%) Forward Only Surviving: 33836 (7.42%) Reverse Only Surviving: 25307 (5.55%) Dropped: 20393 (4.47%) TrimmomaticPE: Completed successfully

Less reads were kept after filtering by quality score equal to 30:
```
python3 -c "print($(cat amp_res_1.trimmed30.fastq | wc -l)/4)"
```
**Output:** 376340.0
```
python3 -c "print($(cat amp_res_2.trimmed30.fastq | wc -l)/4)"
```
**Output:** 376340.0

Run fastqc on modified fastq files trimmed with quality 30 to check how trimming changed the quality of the data:

```
fastqc -o . amp_res_1.trimmed30.fastq amp_res_2.trimmed30.fastq
```

For amp_res_1.trimmed30.fastq test on *per tile sequence quality* still failed but for amp_res_2.trimmed30.fastq resulted in warning (Fig. S4). Minimal *per sequence quality score* is about 30 ( belongs to the green area of quality scores ) which is a little higher than in case of trimming by quality score of 20 (minimal per sequence quality score is about 24 and falls into orange area of quality scores). Test on *sequence length distribution* also raised a warning indicating that sequences vary by length.

**5 . Aligning sequences to reference**

**5.1 Index the reference file**

The standard option for human and mouse would be -a bwtsw, which creates indices based on the Burrows-Wheeler Transform (BWT), but we did not use this key as far as we were dealing with relatively small bacterial genome:

```
mkdir bwa_ouput
bwa index GCF_000005845.2_ASM584v2_genomic.fna -p bwa_ouput/ref
```

What files were generated?

```
ls bwa_ouput
```

**Output:** ref.amb ref.ann ref.bwt ref.pac ref.sa

5 files were generated: with *.sa, *.pac, *.bwt , *.ann and *.amb extension.

**5.2 Alignment of reads**

a) that were trimmed with quality score of 30:

```
bwa mem ref amp_res_1.trimmed30.fastq amp_res_2.trimmed30.fastq >
amp_res.trimmed30.sam
```

b) that were trimmed with quality score of 20:

```
bwa mem ref amp_res_1.trimmed.fastq amp_res_2.trimmed.fastq >
amp_res.trimmed20.sam
```

c) that were not trimmed:

```
bwa mem ref amp_res_1.fastq amp_res_2.fastq > amp_res.sam
```

### 5.3. Compress SAM files

```
samtools view -S -b amp_res.trimmed30.sam > amp_res.trimmed30.bam
samtools view -S -b amp_res.trimmed20.sam > amp_res.trimmed20.bam
samtools view -S -b amp_res.sam > amp_res.bam
```

Getting basic statistics:

```
samtools flagstat amp_res.trimmed30.bam
```

**Output:**

752818 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
138 + 0 supplementary
0 + 0 duplicates
**752200** + 0 mapped (**99.92%** : N/A)
752680 + 0 paired in sequencing
376340 + 0 read1
376340 + 0 read2
750112 + 0 properly paired (99.66% : N/A)
751554 + 0 with itself and mate mapped
508 + 0 singletons (0.07% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)

```
samtools flagstat amp_res.trimmed20.bam
```

**Output:**

892776 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
258 + 0 supplementary
0 + 0 duplicates
**891649** + 0 mapped (**99.87%** : N/A)
892518 + 0 paired in sequencing
446259 + 0 read1
446259 + 0 read2
888554 + 0 properly paired (99.56% : N/A)
890412 + 0 with itself and mate mapped

979 + 0 singletons (0.11% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)

```
samtools flagstat amp_res.bam
```

**Output:**

912095 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
343 + 0 supplementary
0 + 0 duplicates
**910940** + 0 mapped (**99.87**% : N/A)
911752 + 0 paired in sequencing
455876 + 0 read1
455876 + 0 read2
907476 + 0 properly paired (99.53% : N/A)
909488 + 0 with itself and mate mapped
1109 + 0 singletons (0.12% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)

99.92% (752200 ) and 99.87% (891649) of reads were mapped after trimming by quality score of 30 and 20, respectively. 99.87% of reads (910940) were aligned to the reference genome without the prior trimming.

Counting number of forward and reverse reads aligned to the reference:
Firstly we extracted only unmapped reads from .bam file:

```
samtools view -f 4 amp_res.trimmed20.bam > unmapped20.sam
samtools view -f 4 amp_res.trimmed30.bam > unmapped30.sam
```

Next step is to find unique flag values (column 2 in bam format):

```
cat unmapped20.sam | cut -f2 | sort | uniq
cat unmapped30.sam | cut -f2 | sort | uniq
```

**Output (same for both files):**

117
133
141
181
69

77

We examined the meaning of each flag using Decoder of SAM flags (http://broadinstitute.github.io/picard/explain-flags.html) and found, that 69, 77 and 117 mean that this read is the first in the pair (forward); 133, 141, 181 mean that a read with this flag is the second in the pair (reversed). So we count a number of reads with each flag using grep:

```
cat unmapped30.sam | cut -f 2 | grep "117" | wc -l
```

**Output:** 90

We repeated this command with all flag meanings and for both files of unmapped reads. Next we found a sum of reads with flags 69, 77, 117 for each file of unmapped reads (forward reads); and a sum of reads with flags 133, 141, 181 for each file of unmapped reads (reverse reads). Thus we count a number of unmapped forward and reverse reads for both files:

| Unmapped reads | Trimmed with quality 20 | Trimmed with quality 30 |
|---|---|---|
| **Forward** | 221 | 260 |
| **Reverse** | 1054 | 358 |

Finally we subtract a number of unmapped reads from input number of reads for alignment and found total number of aligned forward and reverse reads:

| Mapped reads | Trimmed with quality 20 | Trimmed with quality 30 |
|---|---|---|
| **Forward** | 446038 | 376080 |
| **Reverse** | 445205 | 375982 |

As we can see less reverse reads were aligned than forward (especially in case of reads trimmed with quality 20). This may be due to the fact that reverse reads have lower quality than forward reads.

### 5.4 Sort and index BAM file

Sorting *.bam files:

```
samtools sort  amp_res.trimmed30.bam -o amp_res.trimmed30.sorted.bam
samtools sort  amp_res.trimmed20.bam -o amp_res.trimmed20.sorted.bam
samtools sort  amp_res.bam -o amp_res.sorted.bam
```

Indexing *.bam file for faster search:

```
samtools index amp_res.trimmed30.sorted.bam
samtools index amp_res.trimmed20.sorted.bam
samtools index amp_res.sorted.bam
```

**6. Variant calling**

Creating mpileup files:

a) for SAM file with reads trimmed with quality score of 30

```
samtools mpileup -f GCF_000005845.2_ASM584v2_genomic.fna
amp_res.trimmed30.sorted.bam > mpileup30
```

b) for SAM file with reads trimmed with quality score of 20

```
samtools mpileup -f GCF_000005845.2_ASM584v2_genomic.fna
amp_res.trimmed20.sorted.bam > mpileup20
```

c) for SAM file with not trimmed reads

```
samtools mpileup -f GCF_000005845.2_ASM584v2_genomic.fna
amp_res.sorted.bam > mpileupinit
```

Variant calling:

First, we set minimum variant frequency option equal to 50 % for all types of data (reads that were trimmed with quality score of 30 or 20 and not trimmed reads) and compare obtained results:

a) reads were trimmed with quality score of 30 before mapping:

```
java -jar varscan/VarScan.v2.4.1.jar mpileup2snp mpileup30 --min-var-freq
0.50 --variants --output-vcf 1 > amp_res30_varscan50.vcf
```

**Output:**

Min coverage: 8
Min reads2: 2
Min var freq: 0.5
Min avg qual: 15
P-value thresh: 0.01
Reading input from mpileup30
4640996 bases in pileup file
8 variant positions (6 SNP, 2 indel)
1 were failed by the strand-filter
5 variant positions reported (5 SNP, 0 indel)

b) reads were trimmed with quality score of 30 before mapping:

```
java -jar varscan/VarScan.v2.4.1.jar mpileup2snp mpileup20 --min-var-freq
0.50 --variants --output-vcf 1 > amp_res20_varscan50.vcf
```

**Output:**

Min coverage: 8
Min reads2: 2
Min var freq: 0.5
Min avg qual: 15
P-value thresh: 0.01
Reading input from mpileup20
4641343 bases in pileup file
9 variant positions (6 SNP, 3 indel)
1 were failed by the strand-filter
5 variant positions reported (5 SNP, 0 indel)

c) reads were not trimmed before mapping

```
java -jar varscan/VarScan.v2.4.1.jar mpileup2snp mpileupinit
--min-var-freq 0.50 --variants --output-vcf 1 >  amp_res_varscan50.vcf
```

**Output:**

Min coverage: 8
Min reads2: 2
Min var freq: 0.5
Min avg qual: 15
P-value thresh: 0.01
Reading input from mpileupinit
4641524 bases in pileup file
9 variant positions (6 SNP, 3 indel)
1 were failed by the strand-filter
5 variant positions reported (5 SNP, 0 indel)

Compare variants found for each case: extract 3 columns corresponding to genomic coordinate where the variant occurs, reference allele and alternative allele:

```
grep -v '#' amp_res30_varscan50.vcf | cut -f2,4,5
```

**Output:**

93043 C G
482698 T A
852762 A G

3535147 A C
4390754 G T

```
grep -v '#' amp_res20_varscan50.vcf | cut -f2,4,5
```

**Output:**

93043 C G
482698 T A
852762 A G
3535147 A C
4390754 G T

```
grep -v '#' amp_res_varscan50.vcf | cut -f2,4,5
```

**Output:**

93043 C G
482698 T A
852762 A G
3535147 A C
4390754 G T

Identified SNPs are the same in all analyzed files. The same result will be obtained if we set --min-var-freq as 0.10 (as it was used in recent study - Barbosa et al. 2017):

```
java -jar varscan/VarScan.v2.4.1.jar mpileup2snp mpileup20 --min-var-freq 0.10 --variants --output-vcf 1 > amp_res20_varscan10.vcf
```

**Output:**

Min coverage: 8
Min reads2: 2
Min var freq: 0.1
Min avg qual: 15
P-value thresh: 0.01
Reading input from mpileup20
4641343 bases in pileup file
9 variant positions (6 SNP, 3 indel)
1 were failed by the strand-filter
5 variant positions reported (5 SNP, 0 indel)

```
grep -v '#' amp_res20_varscan10.vcf | cut -f2,4,5
```

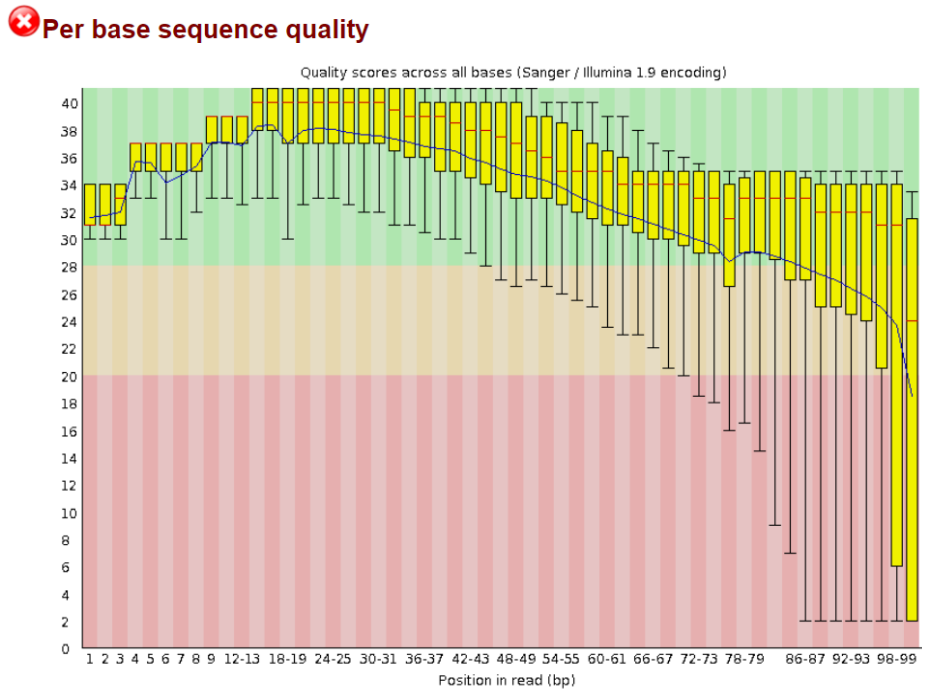
93043 C G

482698 T A
852762 A G
3535147 A C
4390754 G T

**7. Variant effect prediction**

For better understanding and interpretation of IGV results we draw a scheme illustrating an effect of a SNP on DNA, RNA and protein levels (Fig. S4 for the second SNP from Table 1, position 482698). Alignment of reads trimmed with the quality score of 20 and all 5 identified SNPs were visualized using IGV tool. Genomic annotation of reference *E. coli* MG1655 K-12 in GFF format was used as track line. We assumed that all triplets are shown by the IGV tool in the 5'-> 3' direction.

a)




b)

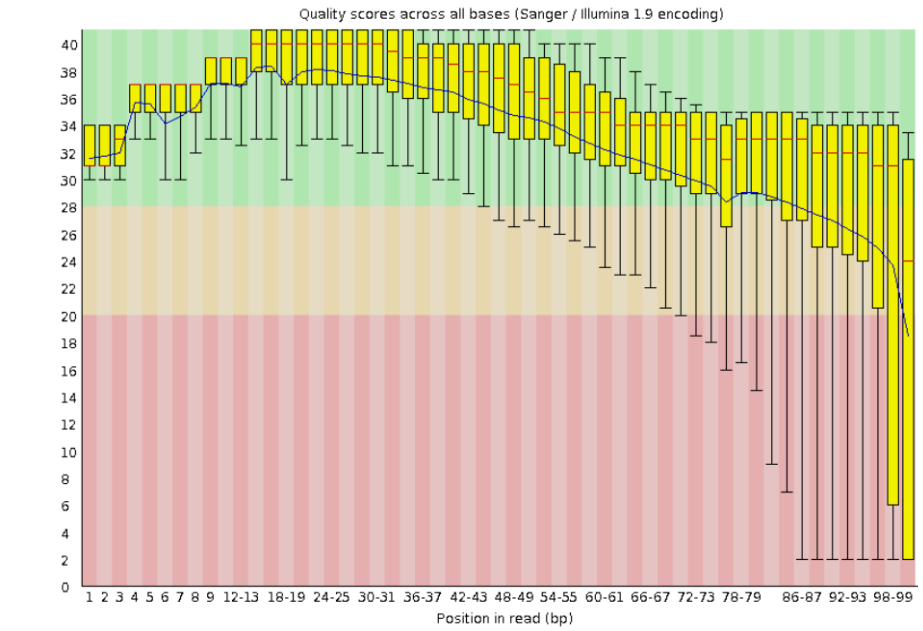



**Figure S1**. Per base sequence quality of raw sequencing data: forward reads (a) and reverse (b) (calculated using FastQC)

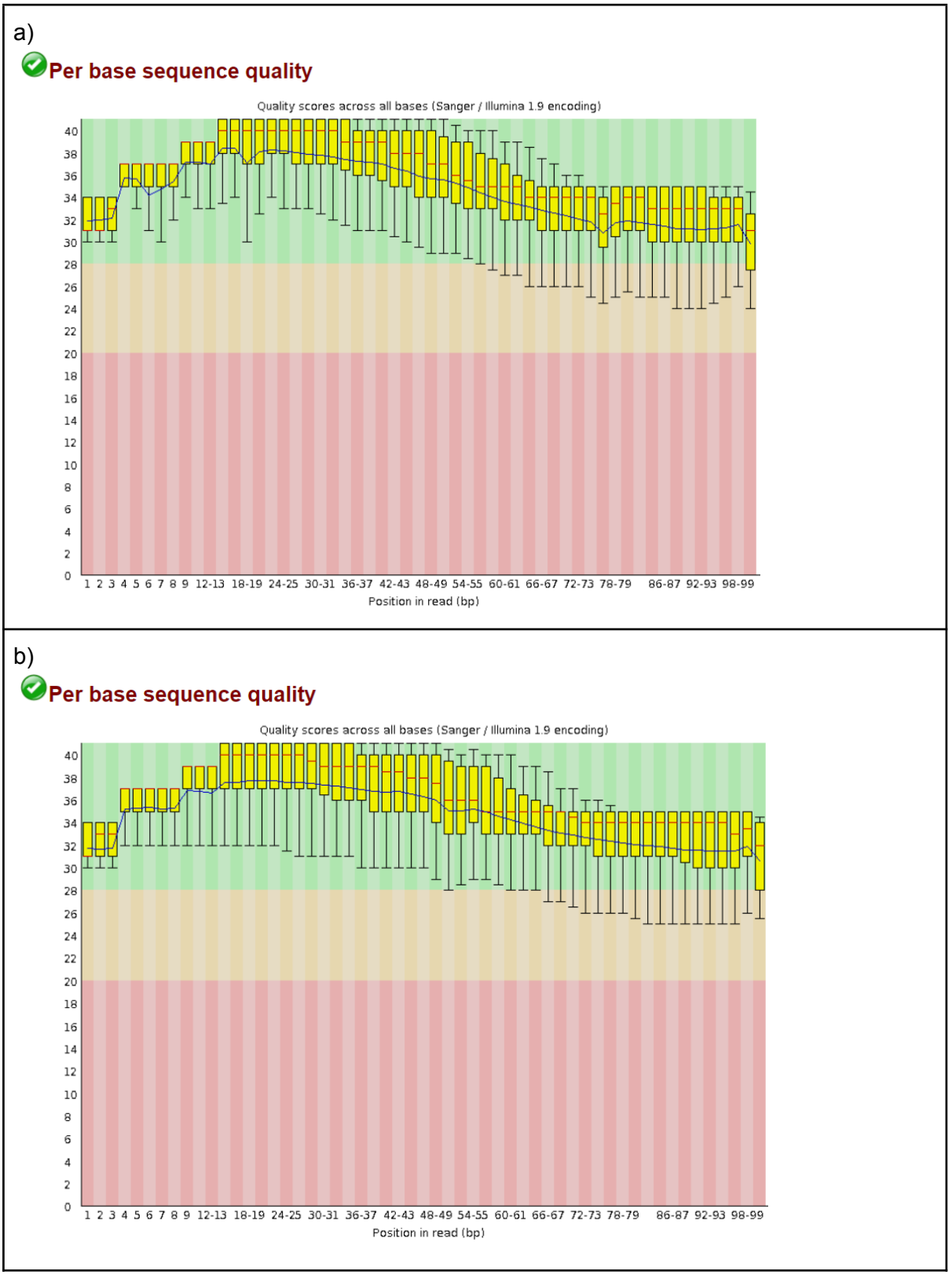
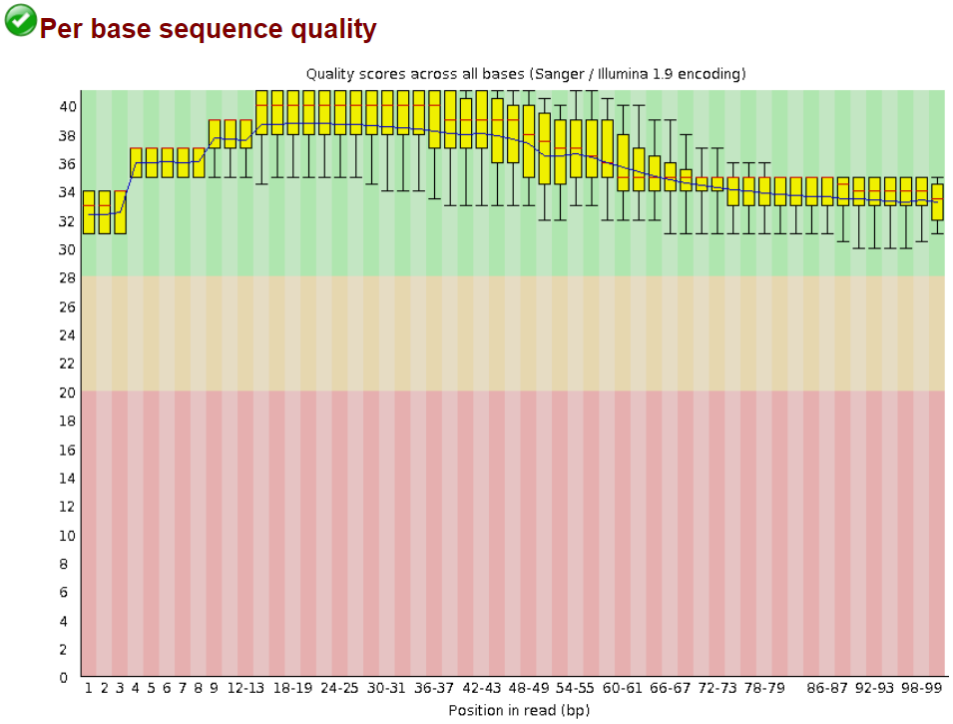



**Figure S2**. Per base sequence quality of sequencing data trimmed with the quality score of 20: forward reads (a) and reverse (b) (calculated using FastQC)

a)




b)

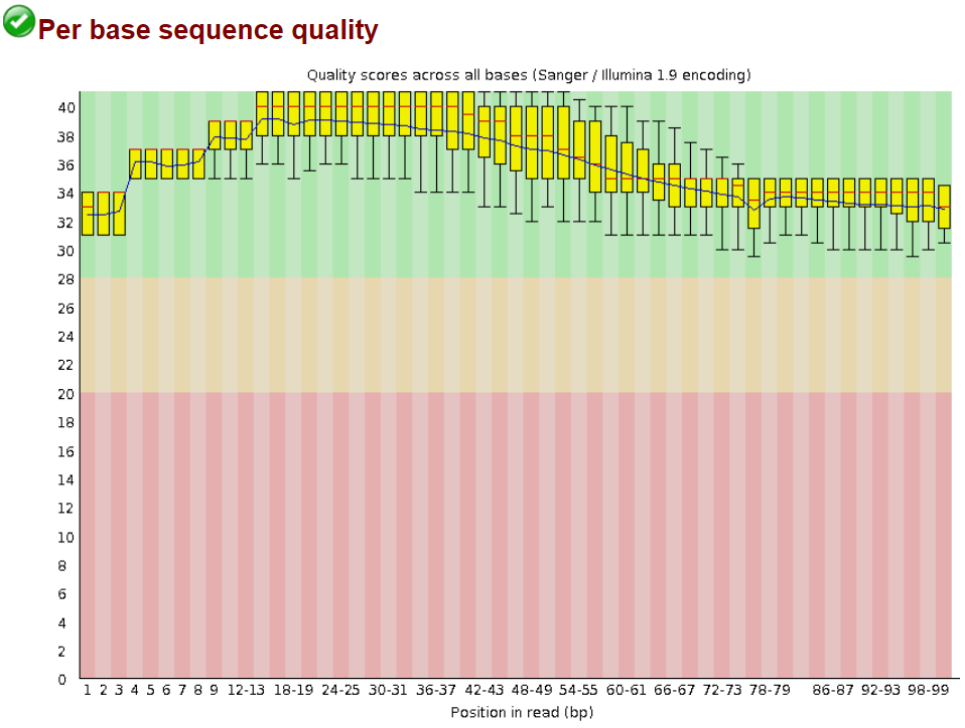



**Figure S3**. Per base sequence quality of sequencing data trimmed with the quality score of 30: forward reads (a) and reverse (b) (calculated using FastQC)

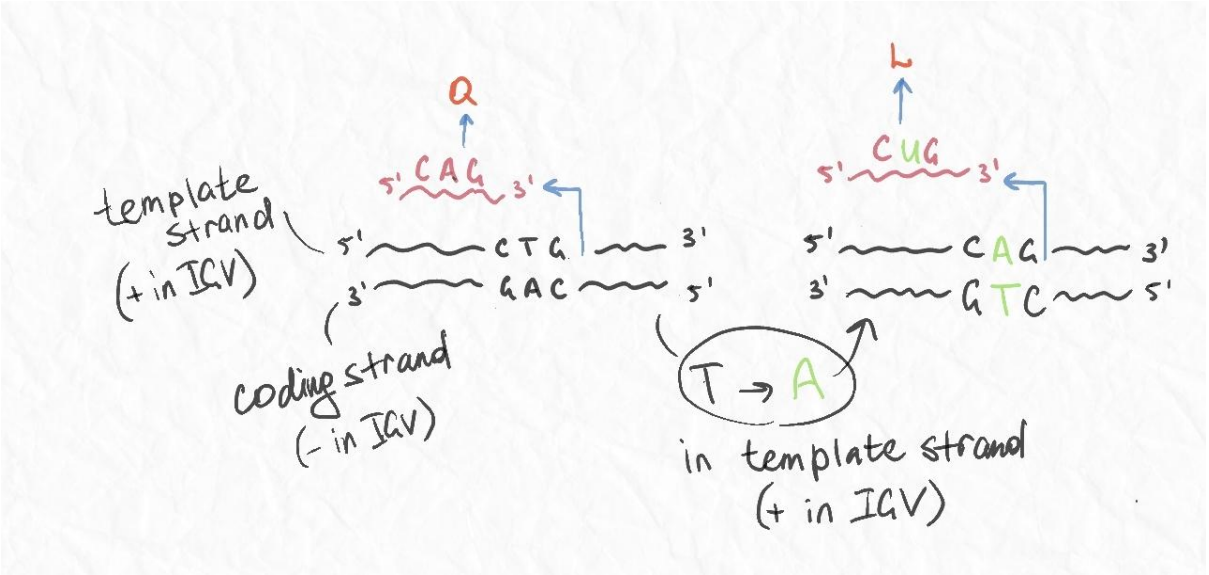



**Figure S4.** SNP effect prediction for the second identified SNP from Table 1 (position 482698). Mutation in template strand (T -> A) results in complementary substitution in another (coding) strand of DNA. Transcribed mRNA contains uracil instead of adenine, which leads to a corresponding missense substitution (Q -> L) in synthesized protein.

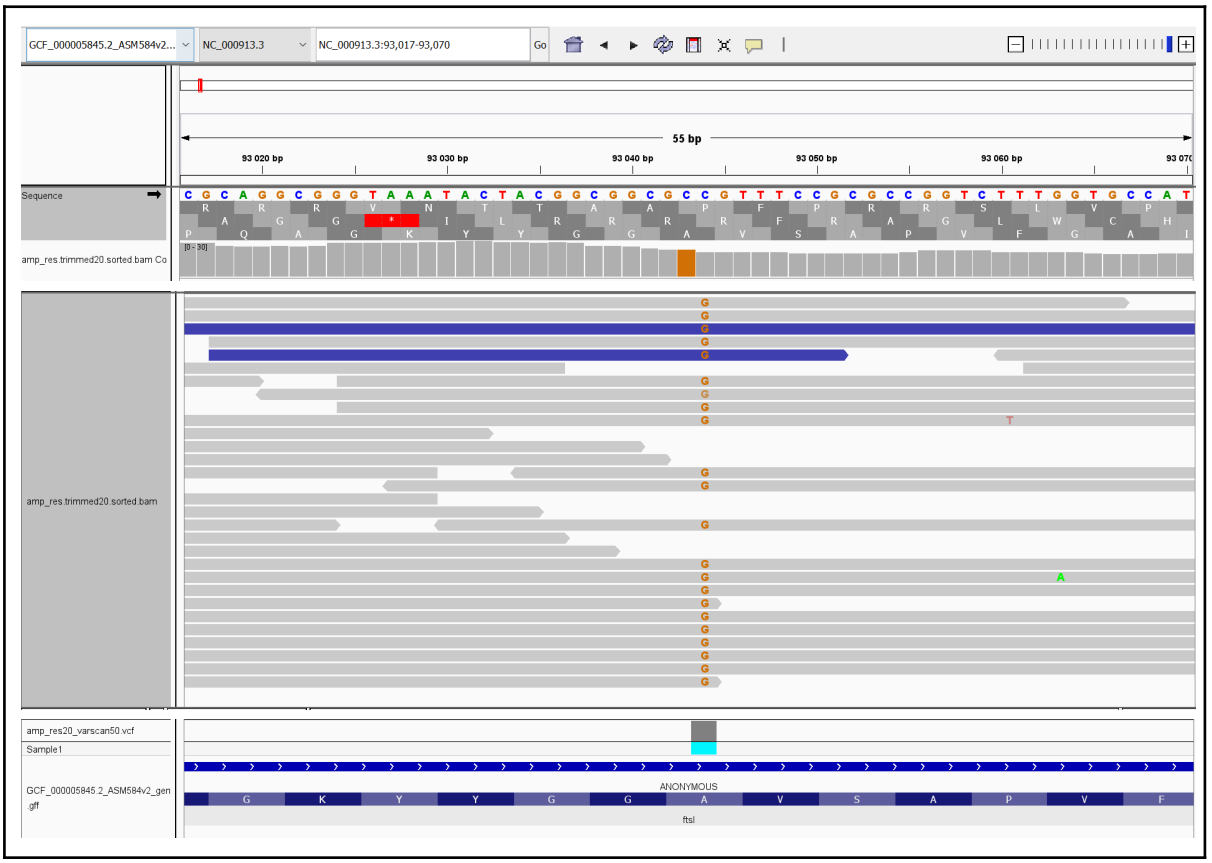



**Figure S5**. SNP at position 93043 on the chromosome. Alignment of reads trimmed with the quality score of 20 and identified SNPs are visualized using IGV tool. Genomic annotation of reference *E. coli* MG1655 K-12 in GFF format is used as track line. Positive strand is shown (sequence arrow points right). In the coding DNA, nucleotide C as a part of triplet GCC was substituted by G resulting in triplet GGC. In mRNA, codon GCC encoding Alanine (A) was changed to GGC encoding Glycine (G).

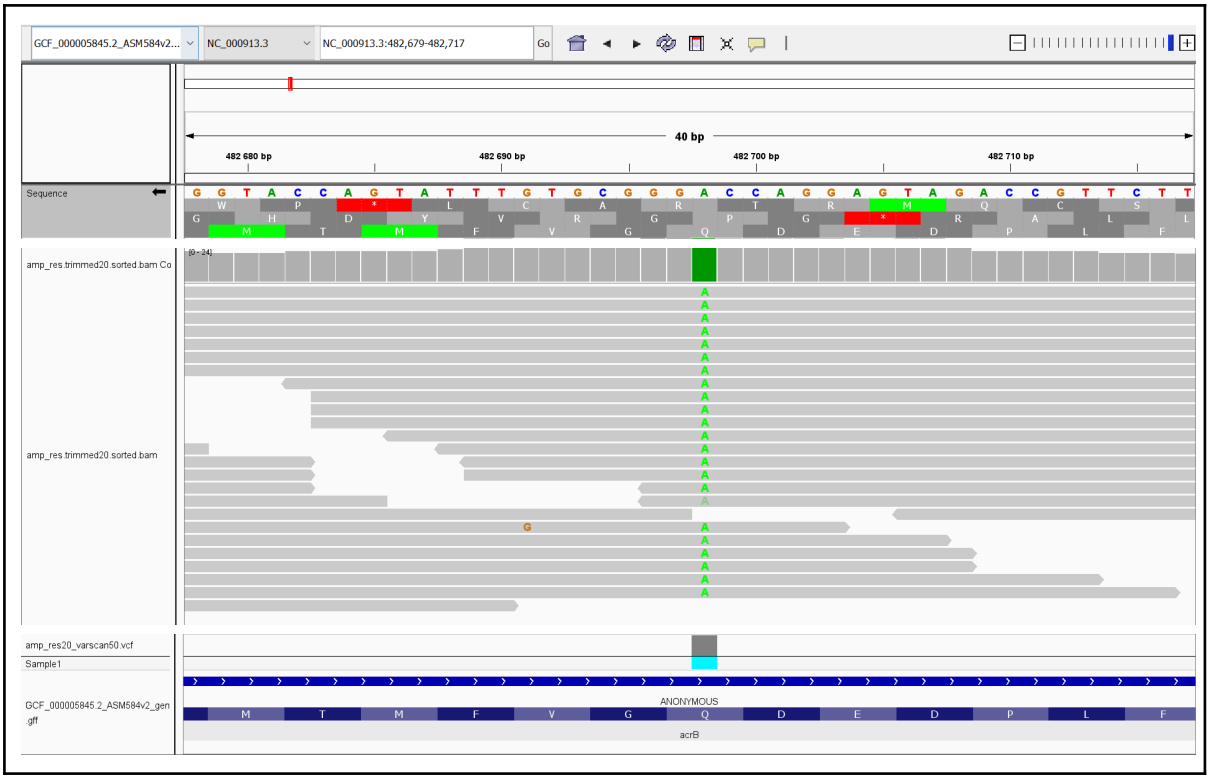



**Figure S6**. SNP at position 482698 on the chromosome. Alignment of reads trimmed with the quality score of 20 and identified SNPs are visualized using IGV tool. Genomic annotation of reference *E. coli* MG1655 K-12 in GFF format is used as track line. Negative strand is shown (sequence arrow points left). In the coding DNA, nucleotide A as a part of triplet CAG was substituted by T resulting in triplet CTG. In the template DNA, nucleotide T was substituted by A. In mRNA, codon CAG encoding Glutamine (Q) was changed to CUG encoding Leucine (L).

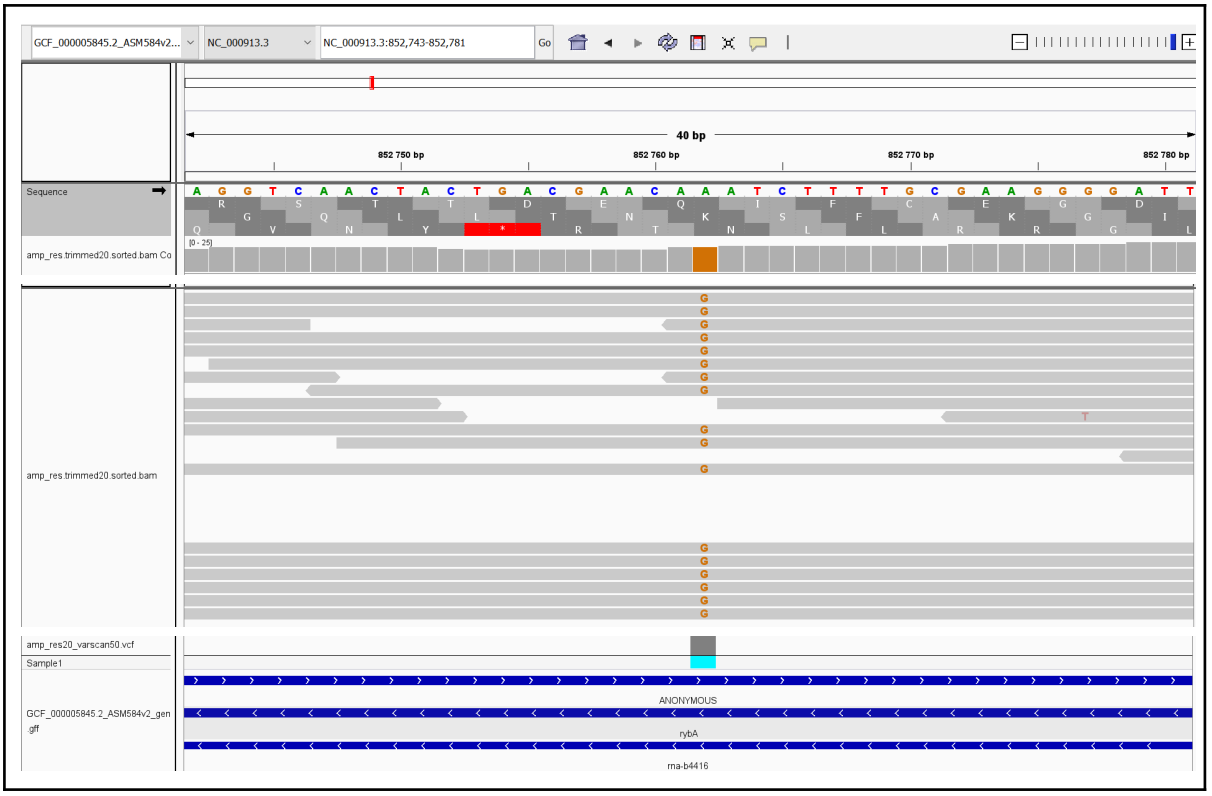



**Figure S7**. SNP at position 852762 on the chromosome. Alignment of reads trimmed with the quality score of 20 and identified SNPs are visualized using IGV tool. Genomic annotation of reference *E. coli* MG1655 K-12 in GFF format is used as track line. Positive strand is shown (sequence arrow points right). SNP occurred in the gene *rybA* encoding small non-coding RNA RybA.


**Figure S8**. SNP at position 3535147 on the chromosome. Alignment of reads trimmed with the quality score of 20 and identified SNPs are visualized using IGV tool. Genomic annotation of reference *E. coli* MG1655 K-12 in GFF format is used as track line. Negative strand is shown (sequence arrow points left). In the coding DNA, nucleotide T as a part of triplet GTA was substituted by G resulting in triplet GGA . In the template DNA, nucleotide A was substituted by C. In mRNA, codon GUA encoding Valine (V) was changed to GGA encoding Glycine (G).

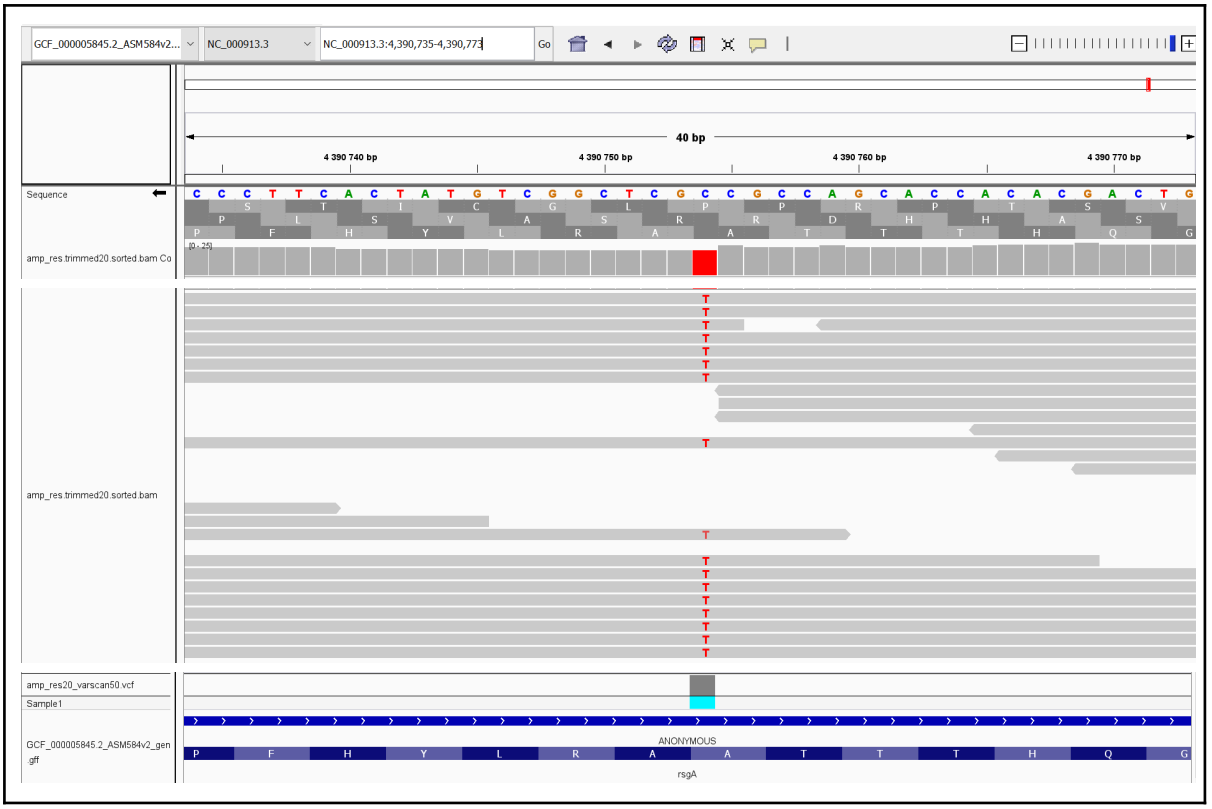



**Figure S9**. SNP at position 4390754 on the chromosome. Alignment of reads trimmed with the quality score of 20 and identified SNPs are visualized using IGV tool. Genomic annotation of reference *E. coli* MG1655 K-12 in GFF format is used as track line. Negative strand is shown (sequence arrow points left). In the coding DNA, nucleotide C as a part of triplet GCC was substituted by A resulting in triplet GCA . In the template DNA, nucleotide G was substituted by T. In mRNA, codon GCC was changed to GCA both encoding Alanine (A).