Identifying the Genetic Basis of Antibiotic Resistance

Authors:

Tejvir Sohi, Alex Nguyen

Abstract:

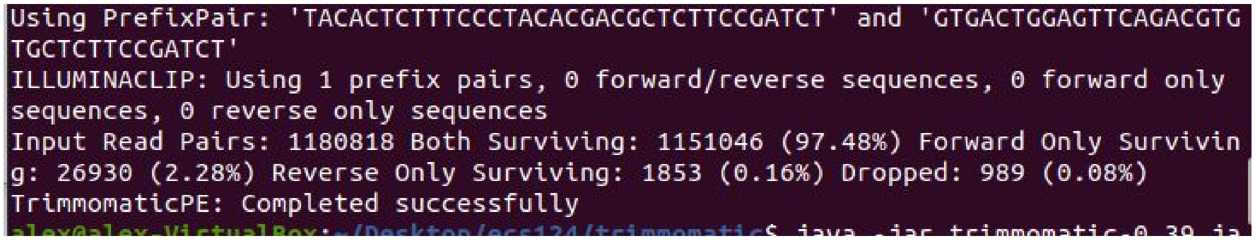
In this project, we compared two different genomes of the bacterium Escherichia Coli with the goal to find any single nucleotide polymorphisms (SNP) that occur between them. The first genome, called the control group, had the E. Coli bacteria living under normal conditions while the second genome, called the resistant group, lived in an environment with constant antibiotic exposure. To find the SNPs, the genomes had to be trimmed to remove low quality portions and increase the quality of the DNA base pairs using trimmomatic. The trimmed files are then properly aligned using bowtie, while also merging the forward and reverse reads of the genomes. After this, samtools is needed to convert to .bcf files that are used by bcftools to prepare proper SNP's for analysis. Bcftools is used to find SNPs for both the control and resistant groups, then the control SNPs are subtracted from resistant SNPs and we are left with the difference. At the end of this process, we had 3 SNPs that had occurred in the resistant group but not in the control group. We cross referenced the SNPs that were produced and checked which genes were impacted the most, with only the first SNP directly affecting a gene, called FTSI (penicillin-binding protein 3). A more detailed explanation of our SNPs can be found under the Results section. Next, we looked into the two biological pathways that were impacted by this SNP. The first is beta-Lactam resistance and the second is peptidoglycan biosynthesis.

Method:

Before we can start working on the project, we need to make sure that we are using the proper operating system and that we can install and run all of the programs. Since running the programs would be best on linux and both of us had windows computers, we agreed to install virtual machine software, specifically Oracle VM VirtualBox, version 1.1.16, and install the 64-bit version of Ubuntu on it. Once this was done, we made sure that the OS had installed all of its updates and drivers so that nothing would interrupt us once we started running the programs.

The first step is to trim out any adapter sequences to improve the alignment accuracy of the reads. We both installed Trimmomatic-0.39 into our virtual machine and started to trim. Our command trims the adapter sequence TruSeq3-PE from control 1 forward and reverse. It outputs the sequence that we want including the trimmed adapters into two separate files called trim and untrim. This was repeated for all of our controls and resistants. In our command we used ILLUMINACLIP, which cuts the adapter sequences from our reads. The command also includes SLIDINGWINDOW, which scans from the 5' end of the sequence clipping any reads with averaging quality below a certain threshold. The LEADING and TRAILING were set to 3, which will cut off the base pairs at both the beginning and the end of the read. Finally we also set MINLEN to 25, which drops any read that falls below this length. The keepBothReads was used in order for trimmomatic to keep both reads in a pair instead of keeping one and removing one and if not done could result in a wrong trimmed sequence. As seen in Figure 1, the original sequence of control 1 forward and reverse 989 read pairs were dropped due to the fact that they were adapter sequences. This command took around an hour to execute in total for all the controls and resistant files. The output was our first proof that the command we used trimmed properly.

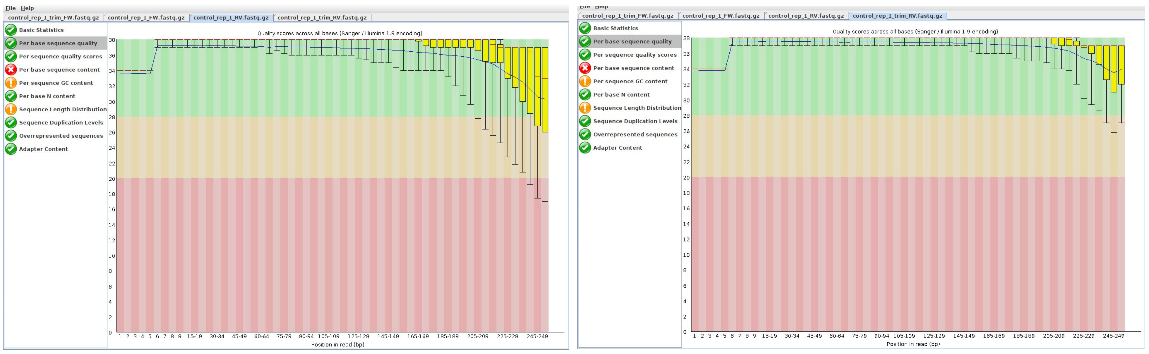
Figure 1:



```
Using PrefixPair: 'TACACTCTTTCCCTACACGACGCTCTTCCGATCT' and 'GTGACTGGAGTTCAGACGTG
TGCTCTTCCGATCT'
ILLUMINACLIP: Using 1 prefix pairs, 0 forward/reverse sequences, 0 forward only
sequences, 0 reverse only sequences
Input Read Pairs: 1180818 Both Surviving: 1151046 (97.48%) Forward Only Survivin
g: 26930 (2.28%) Reverse Only Surviving: 1853 (0.16%) Dropped: 989 (0.08%)
TrimmomaticPE: Completed successfully
alex@alex-VirtualBox:~/Desktop/ecs124/trimmomatic$ java -jar trimmomatic-0.39.ja
```

Just looking at the output was not enough evidence for a high quality read, so we took all of our trimmed files into another program called FASTQC. First, we installed the FASTQC version 0.11.9 and opened all the sequence files to check the per sequence quality scores. From this initial observation we noticed the forward sequences for the controls and resistants mostly fell in good quality range, so trimming it made the quality even better. The bigger issue was our reverse sequence files. These files were mostly in very poor quality, which are unusable for any type of alignment. In Figure 2, we see the original reverse sequence of control 1 on the left, which falls into poor quality before being trimmed, but through trimming TruSeq3-PE the quality falls slightly above the green zone on the right.

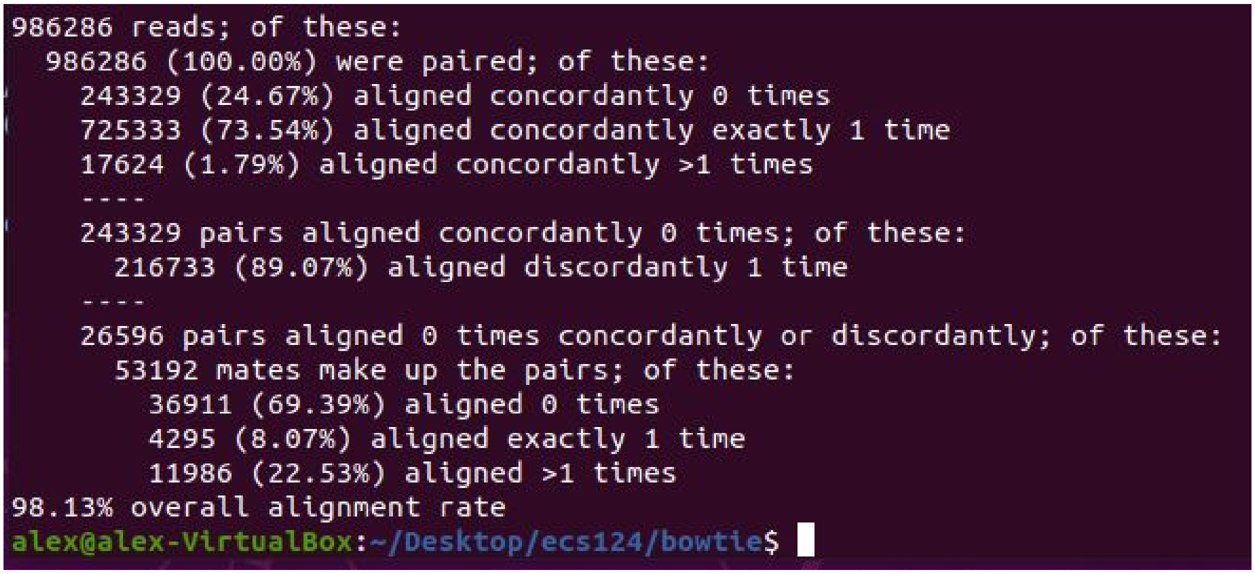
Figure 2:


File Help
control_rep_1_trim_FW.fastq.gz
control_rep_1_FW.fastq.gz
control_rep_1_RV.fastq.gz
control_rep_1_trim_RV.fastq.gz
Basic Statistics
Per base sequence quality
Per sequence quality scores
Per base sequence content
Per sequence GC content
Per base N content
Sequence Length Distribution
Sequence Duplication Levels
Overrepresented sequences
Adapter Content
Quality scores across all bases (Sanger / Illumina 1.9 encoding)
Position in read (bp)

Although further trimming attempts can be made using other adapter sequences, this did not result in any improvements in the per sequence quality scores. Therefore, we decided that the reads were high enough quality after one trimming. We finalized our method of trimming only TrueSeq3-PE from all of the sequences and continued on to the next step, .sam files.

To start creating the .sam files we first installed Bowtie2 version 2.4.2. Then we went to EcoCyc.org and downloaded a .fasta file which included the e-coli genome sequence file. We used this file to create an index file and from this index file, we were able to create .sam files. The .sam file is the alignment of the control and resistant files to the original e-coli sequence. Since .sam files are readable, we tried opening them and reading them to see if there were any noticeable patterns. Without any luck we decided to continue our search for SNPs by creating a .bam file. As seen in Figure 3, the .sam files aligned with an overall 98.13% alignment rate. This is a good sign because both the resistant and sequence files are both e-coli, which should result in a high alignment rate. This command took almost 15 minutes for each control and resistant file resulting in a long wait before our next step to finding SNPs.

Figure 3:



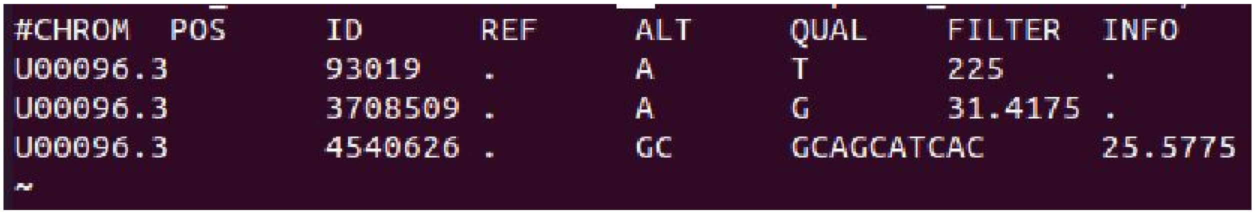
```
986286 reads; of these:
  986286 (100.00%) were paired; of these:
    243329 (24.67%) aligned concordantly 0 times
    725333 (73.54%) aligned concordantly exactly 1 time
    17624 (1.79%) aligned concordantly >1 times
    ----
    243329 pairs aligned concordantly 0 times; of these:
      216733 (89.07%) aligned discordantly 1 time
    ----
    26596 pairs aligned 0 times concordantly or discordantly; of these:
      53192 mates make up the pairs; of these:
        36911 (69.39%) aligned 0 times
        4295 (8.07%) aligned exactly 1 time
        11986 (22.53%) aligned >1 times
98.13% overall alignment rate
alex@alex-VirtualBox:~/Desktop/ecs124/bowtie$ 
```

For the next step, we installed Samtools version 1.11 and created a .bam file for the three control and three resistant files. This .bam file is a binary file and will be used to create the .bcf files needed for the last steps. Another program is needed at this point, which is Bcftools version 1.10.2. The .bcf files are then converted to a .vcf file, but we came across an error where the file could only be detected as a diploid sequence. Since e-coli are haploid we had to add another command, --ploidy 1, which tells bcftools that the file has a haploid genome. After the .vcf files were created we have to index them and create a folder called dir which has files that contain the resistant SNPs. We then index the file called 0000.vcf.bgz. Then we run our last command that deletes the control SNP from the resistant files. Our final product will be the new 0000.vcf.bgz file. This will contain the coordinates of the SNP we were looking for, which we will use in NCBI to find more information about the gene and pathways the gene is involved with.

Results:

Once we finish gathering the SNPs from the genomes, we can open the 0000.vcf.bgz file and at the bottom as seen in Figure 4 there is a list of SNPs that are only present in the genomes of the resistant group of bacteria. From this list, we need to focus on the "ID" column, which contains the number that identifies the position of each SNP is located in the entire sequence.

Figure 4:



```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO
U00096.3        93019   .       A       T       225     .
U00096.3        3708509 .       A       G       31.4175 .
U00096.3        4540626 .       GC      GCAGCATCAC      25.5775
~
```

Using this identifier, we can find relevant genes from NCBI's genome browser, which we also used to find our reference genome. The first SNP directly coded a gene and provides us with two pathways for that gene: beta-Lactam resistance and peptidoglycan biosynthesis. According to NCBI, beta-Lactam resistance is a widely used antibiotic product directly related to "interfering the structural crosslinking of the peptidoglycans in bacterial cell walls" (NCBI).

From this deduction, we can safely say that this was the antibiotic the resistant group was constantly exposed to in the sequence we were given. The second pathway is peptidoglycan biosynthesis, which is a macromolecule formed from "crosslinking short peptides to form the cell wall in bacteria surrounding the cytoplasmic membrane" (NCBI). With both the pathways identified our conclusion for this gene is that the antibiotic resistance worked against the biosynthesis in order to kill the peptidoglycan. Instead of killing it, through the constant exposure it became more resistant to beta-lactam resistance by building stronger cell walls due to the evolving peptidoglycan biosynthesis.

Looking further into the next two SNPs we found they were not directly coding for any gene, so instead we decided to increase the range and find the next closest gene that it could have possibly affected. Our new ranges listed below gave us the results of proK (type of tRNA) and fimB (regulator for fimA). In conclusion, only our first SNP is coding a gene, with the other two SNPs having no direct gene mutations.

SNP Positions	Mutated Genes	Pathways for the Gene
93019	fstI (Peptidoglycan DD-Transpeptidase)	- Beta-Lactam Resistance - Peptidoglycan Biosynthesis
3708509 (Range: 3708309 to 3708709)	N/A (proK: tRNA)	N/A
4540626 (Range: 4540626 to 4541626)	N/A (fimB: Regulator for fimA)	N/A

Author contributions:

We split up the work evenly, with Tejvir handling the control sequence files and Alex managing the resistant sequence files. Alex finished merging the sequences and printing out the final result and Tejvir completed the analysis of the SNPs.

Work Cited

“Beta - Lactam Resistance - BioSystems - NCBI.”*NCBI*, www.ncbi.nlm.nih.gov/biosystems/1060015?Sel=geneid:944799#show=genes. Accessed 8 Dec. 2020.

“Escherichia Coli Str. K-12 Substr. MG1655, Complete Genome - Nucleotide - NCBI.” *NCBI*, www.ncbi.nlm.nih.gov/nuccore/U00096.3?report=gbwithparts&log$=seqview. Accessed 8 Dec. 2020.

“Peptidoglycan Biosynthesis - BioSystems - NCBI.” *NCBI*, www.ncbi.nlm.nih.gov/biosystems/1059?Sel=geneid:944799#show=genes. Accessed 9 Dec. 2020.