Salmonella Outbreak

Mattia Delleani, Marco Fioretti, Francesca Paoletti

December 14, 2021

1 Introduction

The salmonella outbreak refers to a violent bacterial outbreak that is currently happening, killing tons of people, and the usual antibiotics have absolutely no effect. Hopefully, Emmanuelle Charpentier, an oustanding biologist, managed to isolate two different strains of the bacteria: the resistant strain to the tetracycline and a wild type.

The Transatlantic Taskforce on Antimicrobial Resistance (TATFAR) asked us to provide a computational tool which is able to determine the origin of the problem, and possibly give biologists some insights on what could be done.

The code is available at: code

2 Data

The tool [4] developed is able to take as input two different FASTA files and give in output a list of SNPs, which are a germline substitutions of a single nucleotide at a specific position in the genome.

The input files are stored in the FASTA format, which is a text-based format for representing either nucleotide sequences or amino acid sequences, in which nucleotides or amino acids are represented using single-letter codes.

The input files contain the sequencing data of the bacterium: the first one consists of the reads of the wild type, which is the phenotype of the typical form of a species as it occurs in nature, which can be killed by antibiotics while the second contains the variant type, which is resistant to antibiotics.

The sequencing operation is performed using the Illumina sequencing method, which is showed in Figure 1.

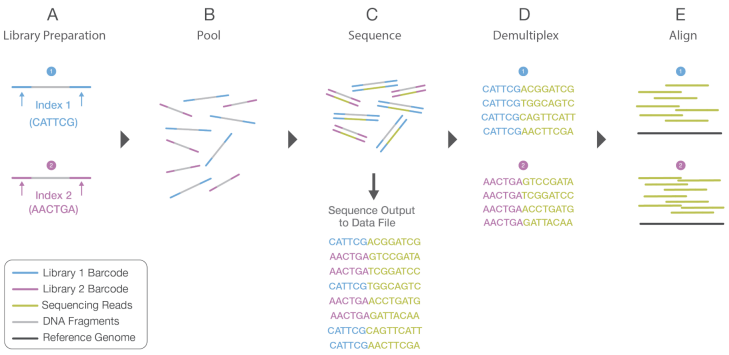


Figure 1: Illumina sequencing method.

3 Implementation

3.1 Statistical model: Poisson

Illumina technique generates reads of $L = 250bps$ and allows to have only a local view of the genome. Even if it is a reliable sequencing method, a small error (1%) is always performed on data. The first task is therefore to detect errors in the sequences of the first file and to remove them, before computing the analysis.

Both files, *Wild Type* and *Variant*, have a dimension $G = 5Mbps$, with $N_w = N_v = 1993167$ number of reads, that are randomly distributed along the genome. Taking one random read, the chance that it is overlapping the position i is:

$$P = \frac{L}{G - L + 1}$$

The number of reads overlapping position i, X_i, can therefore be described with a *Binomial Distribution*:

$$X_i \sim Bin(N, \frac{L}{G - L + 1})$$

If we consider small sets of K elements (k-mers), the probability that a read overlapping a k-mer in position i is

$$P = \frac{N - k}{G - L + 1}$$

and the number of reads overlapping a k-mer in position i can be described again by a *Binomial Distribution*.

$$X_i \sim Bin(N, \frac{N - k + 1}{G - L + 1})$$

If the number of reads N is really high and the length of each read is small, the number of times a k-mer **s** is repeated in the reads can be described by a *Poission Distribution*.

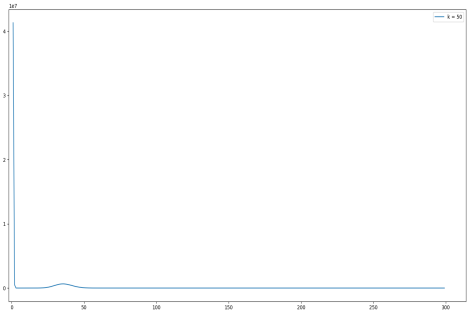
$$Y_s \sim Poisson(N * \frac{L - k + 1}{G})$$

If Y_s is very small, the number of repeated k-mers in the reads is small, so it's likely to have a sequencing error.

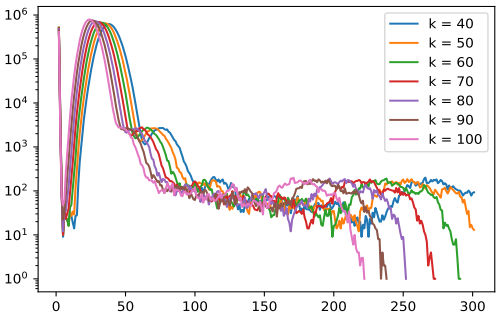
3.2 Estimate sequencing error

In order to detect the sequencing errors present in the file, for each read encountered, we associate its total number of occurrences. Actually, the objective is to find a Poisson distribution, which means to find an adapted window size such that the mean of occurrences is near to the number of covers in each nucleotide. To better understand the problem it is better to look at the following graphs (Figure 2a and Figure 2b), where on the x-axis we have the multiplicity of the k-mers present in the genome and on the y-axis, which is in a logarithmic scale, the number of distinct k-mer that appeard in the genome with the same multiplicity, called abundance [2].

As we can see in Figure 2a, there is an initial sharp peak that cannot allow us to understand properly the trend of the plot and neither to allow us to proceed with the recruitments made previously. This means that most erroneous k-mers occur only once in the reads and the cardinality of those with multiplicity two or more, that are significantly lesser as observed Figure 2a and Figure 2b, are our objects of intrest. So, we can assume that erroneous k-mers



(a) K-mer abundance histogram of reads with $k = 50$. x-axis corresponds to multiplicity values and y-axis corresponds to k-mer counts.

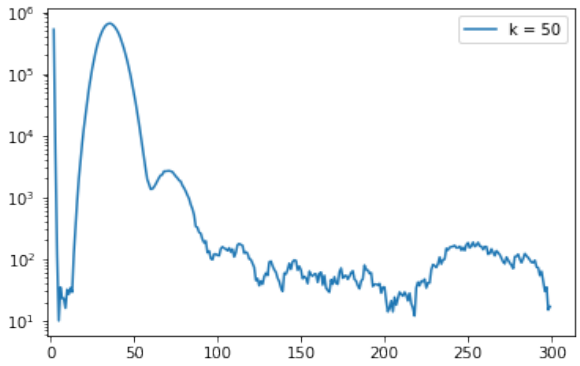


(b) K-mer abundance simulation, with different value of K

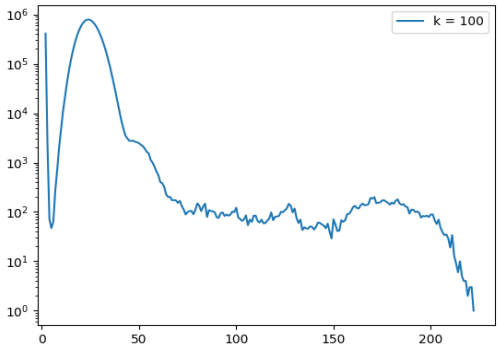
result from single position errors that contribute to an initial sharp peak, thus thay can be discarded by our model. Once removed those k-mers, we have plotted the distribution of our data for different k values, see Figure 2b, and as before, on the x-axis we have the multiplicity of the k-mers present in the genome and on the y-axis, which is now in a logarithmic scale, the abundance with the initial sharp peak at position 1 removed. The value of k need to be sought empirically taking into account two factors:

- too small will lead to many short contigs resulting in higher memory usage.
- whereas a longer kmer will lead to few long contigs and lower memory usage.

Furthermore, if the choice of k is too low, the probability of concatenation increases and as a result of the programme we obtain numerous sequences of which only a few are SNPs (see section on concatenation). We have seen empirically that by choosing values between $25 \leq k \leq 100$ results are the same. All examples given from now on will consider $k = 50$.



(a) K = 50.



(b) K = 100.

Now, considering the mean frequency of occurrence of all the sequencies, it's likely that k-mers whose frequency is lower than a given value may be considered as errors made during the sequencing operation. Therefore, once the k-mers with frequency 1 have been removed, it is easy to see from the graph that other frequencies around one can also be considered as errors. In fact, after displaying the plots, the program requires to introduce an input parameter ($p \in N^+$) that specifies where to start considering the k-mers as valid and not as sequencing errors. Ideally, the value of p should be sought as the minimum of the convexity of the function

between the origin of the axes and the maximum peak of the frequency. Since the numbers of occurrences of k-mers on the x-axis are integers, and the function is not continuous, for simplification we introduce the value as an input parameter. After many attempts, we can say that empirically the ideal value is found around $p = 10$. Once p is introduced, all the k-mers with number of occurences less than this value are dropped out from the dictionary and the remaining are used to continue the analysis. This task is perofrmed for both files and as resullt we obtain two data structures (one for the wild and one for the variant) containing the k-mers without the sequencing errors.

3.3 Genome reconstruction and SNPs detection

After having found the best value for the window size and removed the sequences errors, the objective is to recognize mutations, comparing the remaining sequences of the two files. To find out the mutations, first we have removed all the sequencies of the first file that appear at least once in the second and viceversa, because if a sequence belonging to the first file (wild-type) is identical to one in the second file (variant), this means that there have been no mutations in this sequence and therefore no SNPs are present. As a result of this procedure we obtain two data structures containing k-mers which are only present in one of the two data structures, so mutations may have occurred. Then, for each data structure, the programme reconstructs part of the genome using the remaining k-mers. For each k-mer, the programme tries to concatenate another k-mer that differs from the first only by a character in the tail or in the head, if this is found then a string is constructed that corresponds to the concatenation of the two k-mers and, to speed up the search, the two k-mers are removed from the data structure and the new sequence obtained is added. For example, as shown in Figure 4, at Phase 1 given two starting k-mers, ACCTGA and CCTGAT, for which the concatenation is possible, a string of length k+1 is generated (ACCTGAT) and the two used k-mers are removed from the data structure. From the next phase (in the example Phase 2) the concatenation is searched between the string generated in the previous phase (if any) and another k-mer, in the example CTGATG. Thus, if the concatenation is successful, at each phase the string increases in length by one character. If the concatenation is not possible, the process continues with all the remaining k-mers in the data structure. Once we have obtained

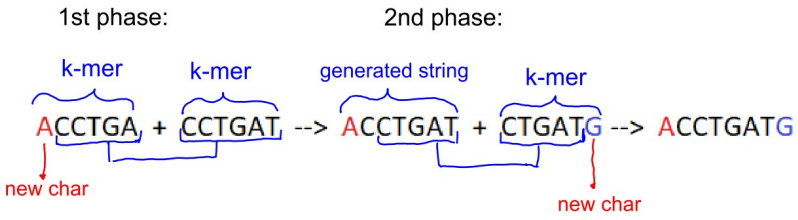


Figure 4: Concatenation process

the genome concatenations for the wild and variant types we can proceed to compare these in order to recognise and be able to visualise the presence of possible SNPs. So, in order to do that, for each sequency in the wild type file, we have computed the distance with respect to each variant sequency, using the Levenshtein measure, which is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

Since a single nucleotide polymorphism (SNP) represents a difference in a single DNA

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\left(\operatorname{tail}(a), \operatorname{tail}(b)\right) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} \operatorname{lev}\left(\operatorname{tail}(a), b\right) \\ \operatorname{lev}\left(a, \operatorname{tail}(b)\right) \\ \operatorname{lev}\left(\operatorname{tail}(a), \operatorname{tail}(b)\right) \end{cases} & \text{otherwise,} \end{cases}$$

Figure 5: Levenshtein distance.

building block, called a nucleotide (for example, a SNP may replace the nucleotide cytosine (C) with the nucleotide thymine (T) in a certain stretch of DNA), for each wild k-mer, we get the minimum distance with respect to the variant k-mer. After this step, for each wild k-mer we get the closest variant sequency, which is likely to be the same sequency in which the mutation happened.

For low values of k ($k < 25$) the programme can generates numerous strings, many of which are not SNPs. Therefore, a filter was inserted to compare strings of the same length and only the closest string in terms of distance is presented as a result.

4 Results

In this section we report the results obtained with $k = 50$ and $p = 10$. As can be seen in the Figure 6, where W stands for *Wild type* and V stands for *Variant*, we obtain two nucleotide sequences whose difference, highlighted in red, is given by a sequence of 3 nucleotides where, observing the first result, thymine is exchanged with guanine (T)→(G) and in the second adenine is exchanged with cytosine (A)→(C). As can easily be seen, the nitrogenous bases have not been exchanged for their complementary ones, so we can say that a mutation has occurred (in this case 3 SNPs in sequence). Using the BLAST[1] software and performing a nucleotide-to-protein search, it is possible to identify these strings as belonging to the TetR family of regulators (TFRs)[3]. TetR, which is short for Tet Repressor proteins, are proteins responsible for the regulation and induction of tetracycline resistance which is involved in regulating antibiotic resistance an important property common to large categories of bacterial species.

```
Printing results:
SNP
        W: TTACATGCCAATACAATGTAGGCTGCTCTACACCTAGCTTCTGGGCGAGTTTACGGGTTGTTAAACCTTCGATTCCGACCTCATTAAGCAGCTCTAATGCG
        V: TTACATGCCAATACAATGTAGGCTGCTCTACACCTAGCTTCTGGGCGAGGGGACGGGTTGTTAAACCTTCGATTCCGACCTCATTAAGCAGCTCTAATGCG
                Distance: 3
-----------------------------------------------------
SNP
        W: CGCATTAGAGCTGCTTAATGAGGTCGGAATCGAAGGTTTAACAACCCGTAAACTCGCCCAGAAGCTAGGTGTAGAGCAGCCTACATTGTATTGGCATGTAA
        V: CGCATTAGAGCTGCTTAATGAGGTCGGAATCGAAGGTTTAACAACCCGTCCCCTCGCCCAGAAGCTAGGTGTAGAGCAGCCTACATTGTATTGGCATGTAA
                Distance: 3
-----------------------------------------------------
```

Figure 6: List of SNPs obtained with $k = 50$ and $p = 10$.

5 Conclusion

In conclusion, this clear and easy-to-use program is able to correctly identify the presence of a mutated gene by satisfying the constraint of having at most linear computational complexity and thus providing an answer to the problem posed by TATFAR.

References

[1] Basic Local Alignment Search Tool, BLAST

[2] Naveen Sivadasan, Rajgopal Srinivasan, Kshama Goyal, *Kmerlight: fast and accurate k-mer abundance estimation*, 2016

[3] Leslie Cuthbertson, Justin R. Nodwell, *The TetR Family of Regulators*

[4] Code available: https://gitlab.ensimag.fr/delleanm/computational-biology-salmonella-outbreakmodal-upload-blob