Project Report for Computational biology

Komarov Artem
Komarov Maksym
Liapustina Mariia
MSc students

Department of Science in Industrial and Applied Mathematics

October 18, 2022

Contents

1 Denotations

- **r** - read length
- **N** - number of reads
- **n** - number of symbols in a FASTA files ($n \approx r \cdot N$)
- **k** - length of k-mer
- **ε** - probability of the nucleotide reading error
- **k-mer** - see section 3
- **FASTA format** - text-based format for representing either nucleotide sequences or amino acid (protein) sequences (in this work nucleotide sequences is used)

2 Formulation of the problem

As part of the project, it is necessary to solve the problem of searching for mutations in the genome.
To search for mutations, there are two simple FASTA sequencing files with Illumina reads, on the basis of which we need to conduct a genome research. Also, during the search for mutations, it is necessary to ignore errors in reading chains of reads.

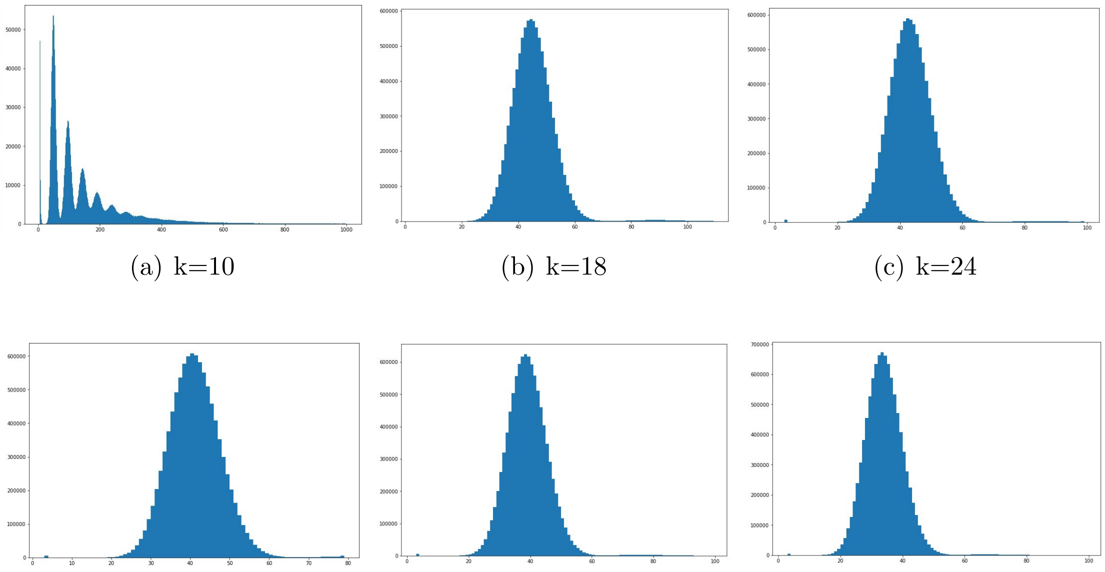
3 Calculation of k-mer

The first stage of the project is to determine the value of k-mer.

K-mers are substrings of length k contained within a biological sequence. Primarily used within the context of computational genomics and sequence analysis, in which k-mers are composed of nucleotides (i.e. A, T, G, and C), k-mers are capitalized upon to assemble DNA sequences.

To determine the optimal value of k, we will take various values for k-mers in the range from 10 to 60. Based on the selected value, a dictionary of occurrences of various k-mers of a given length in reads from both files (origin and variant) is built. After that we paint histograms for origin dictionary, and the resulting histograms should match as closely as possible with the Poisson distribution.

The resulting histograms for $k = [10, 18, 24, 32, 40, 60]$ are following:



(a) k=10

(b) k=18

(c) k=24

(d) k=32

(e) k=40

(f) k=60

If we use a value $k = 10$, then the histogram does not match with the Poisson distribution. If we choose a value $k > 50$, then the chance of error in k-mers increases. But the main problem for the big k is dict memory usage. k has been chosen as the highest one, but with the memory consumption that can afford 16GB memory computational hardware.

4 Errors clipping

For the $k = 32$ the number of possible k-mers is $kMerPossible = 4^k = 4^{32} \approx 10^{19}$. The number of k-mers in a dictionary is $dictLen \approx 5 \cdot 10^8$. Therefore a probability that read error in a k-mer will match another existing k-mer is $p_{match} \approx \frac{dictLen}{kMerPossible} \approx 10^{-10} \ll 1$. Therefore k-mers with read errors will appear in the beginning of the graph (see Figure 1) with the number of occurrences about 1. So such k-mers are clipped (not used for mutations finding).

5 Search for mutations

To search for mutations, we look for unique values in the first dictionary based on origin file that are not present in the second dictionary based on variant file.

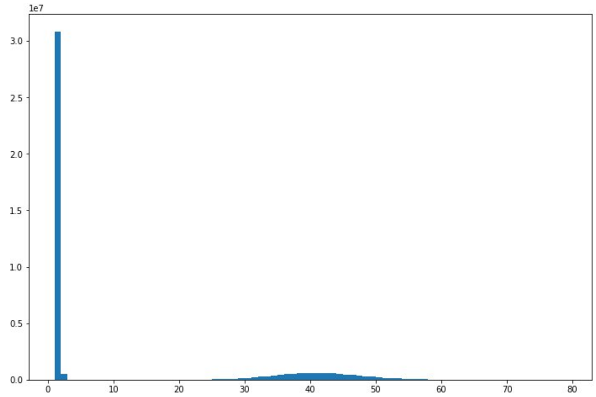


Figure 1: Errors and mutation histogram

We have mutation in each k-mer. Mutation length is less then k-mer / 3, so it can not be in the first and third parts of the k-mer (in the same k-mer). Then we find k-mer from the second file with the same part of the genome (mutation in the same position).

The DNA polymer has a rather complex structure. Nucleotides are linked together covalently into long polynucleotide chains. These chains in the vast majority of cases are combined in pairs with the help of hydrogen bonds into a secondary structure, called the nucleic acid double helix. Since DNA has such a cohesive structure, when a mutation occurs in one polynucleotide chain, the mutation will also be present in the corresponding nucleotide of the second polynucleotide chain.

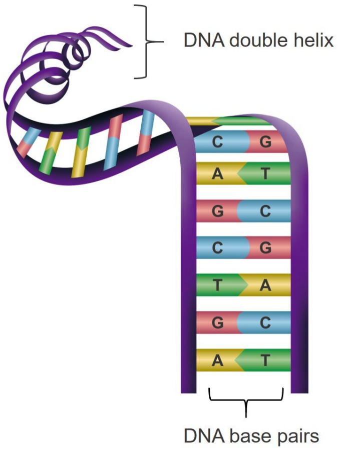


Figure 2: DNA structure

6 Algorithm complexity

1. **k finding and k-mers dict building**
   During k-mers dictionary building the whole reads file is processing one time, each letter (nucleotide) is read only once, therefore for this step the complexity is $O(n)$. During finding proper k constant number of tries has been carried out, therefore complexity is still $O(n)$.

2. **k-mers dicts comparison**
   Number of k-mers in a dict less than $(r - k) \cdot N < n$. For each key in a dict constant time operations are performed - finding element in a dict, which is hash-table. Therefore complexity is $O(n)$. Second dict processed in the same way ⇒ complexity for this step is $O(n)$.
3. **finding mutations using dicts difference**
   Only less than $mutationsNumber \cdot k \cdot 2$ string comparisons required ⇒ complexity for this step is $O(C)$.

For the described steps computing complexity is $O(n)$ or less, therefore overall complexity of the algorithm is $O(n)$.

7 Results

Using model based on the k-mers finding and comparison, we compared two simple FASTA sequencing files and founded SNPs. We proved a linear performance complexity of the model.
Code with some comments you can find on the github.
Resulting mutation pairs:
TAGGCTGCTCTACACCTAGCTTCTGGGCGAGTTTACGGGTTGTTAAACCTTC...
TAGGCTGCTCTACACCTAGCTTCTGGGCGAGGGGACGGGTTGTTAAACCTTC...
—

TGAGGTCGGAATCGAAGGTTTAACAACCCGTAAACTCGCCCAGAAGCTAGGT...
TGAGGTCGGAATCGAAGGTTTAACAACCCGTCCCCTCGCCCAGAAGCTAGGT...