Joan Alcaide Núñez
11 June 2023
Contact: joanalnu5@gmail.com

Protein biosynthesis and DNA mutations

Python code to learn about genetics

Introduction

The last year I attended a biology and genetics introductory course at the Universitat Autònoma de Barcelona (UAB) as part of the university's summer programs for high school students (school years 9, 10, 11 and 12). The course gave me a deep overview of how the genome works and how geneticists collect and analyze data. We even had the opportunity of having a practice day in a lab at the faculty. Here's a link to the summer course, which will be repeated this year. Besides that, this year I have been following a python coding course that imparted Leagues of Code (LoC), which lasted 10 weeks with theory and exercises. With this new knowledge and with the intention of extend my experience on scientific research I coded a program that translates DNA into amino acid chains, generates random mutations and draws statistics from the results.

The code's functions

The code must be saved as '.py' file locally in a computer. In order to run it, one has to have downloaded the python IDLE environment from the official web. Note that the 'DNA Bio' file is the main program while the 'DNA Bio data production' file has some changes in order to run the experiment. Now I will explain how the main code works.

First of all, the human must type (or paste) a DNA sequence, the program will check that this string is correct by looking at some parameters as the first and last bases and the length. The code will translate the DNA into an amino acid chain by a 2 step process. Imitating a natural cell, the program translates the DNA snippet into mRNA. This is done by going along the complete snippet and creating a new string with the mRNA complementary bases of the DNA bases. Thymine is also changed for Uracil. After completing the transcription it follows the translation. Now the mRNA will be read by codons, this mean in

threes. The first codon must encode the amino acid methionine and then the next amino acid is read and inserted in a new string. The program has a complete data base of all 64 possible codons (4^3) and the 20 possible amino acids. By a set of comparisons the amino acid chain is generated. The final results is a single string with the acronyms of the amino acid in the original order of the snippet. This represents the structure of the final protein and the protein biosynthesis is completed. This part of the code was tested with a DNA sequence of the 10A Biology OneNote (page 5). The first half of the code achieves the translation from DNA to amino acid is very much faster than a person would do. Furthermore it enables the study of large numbers of mutations as will be explained later. The code uses a 2 step process, this increases the probabilities of errors, however it similarly illustrates the transcription and translation processes in a natural cells.

Experiment simulation of mutations

With this capability of analysis of DNA sequences an experiment was conducted. The experiment consisted of the generation of 1,000,000 (1 million) random[1] mutations. A set of 10 test cases were conducted, this means that 10,000,000 mutations and 1,413.8 MB of data were generated, what needed more than 24h of computing. The experiment was run on an isolated computer with the 'DNA Bio data production' file, which runs 10 times 1,000,000 times the following code.

The second part of the code is the generation of random mutations and the analysis of these. First the code, changes up to 1 base of the previous provided DNA sequence. The change is selected randomly by a python function called 'randint()'. After mutating the DNA the code runs the transcription and translation of the mutated sequence and the final 'mutated' amino acid chain (protein) is obtained.

In the code it is saved what type of mutations are generated, this enables the draw of conclusions and statistics. At the end, after creating 1,000,000 mutations, the code prints the statistics of the specific test case that were copied manually to an Excel ('analysis DNA mutations from rows 3-7 columns B-K'). The rest of the values are generated by functions that show different proportions and percentages.

Results

[1] For the randomness the randint() function of the random library integrated with python was implemented

Two specific results were considered. On one hand the category of misssense mutations, that represent all the mutations that have generated some change to the final protein or that prevented of the protein biosynthesis. With the comparison of the proteins encoded by the mutations with the original protein this pie chart (fig. 1) could be generated. The chart shows the percentages of mutations that affected the final protein, misssense mutations (in red) and the ones that were not affected (in green). This means that about 3 in 4 mutations affected the final development of the protein and didn't permit the cell to accomplish its function completely or correctly[2]. However it is important to remember that the malfunction of one single cell doesn't affect the vast majority of living species like humans. However it is interesting to see how fragile a living being is and how important it is to us an appropriate conservation of the DNA.

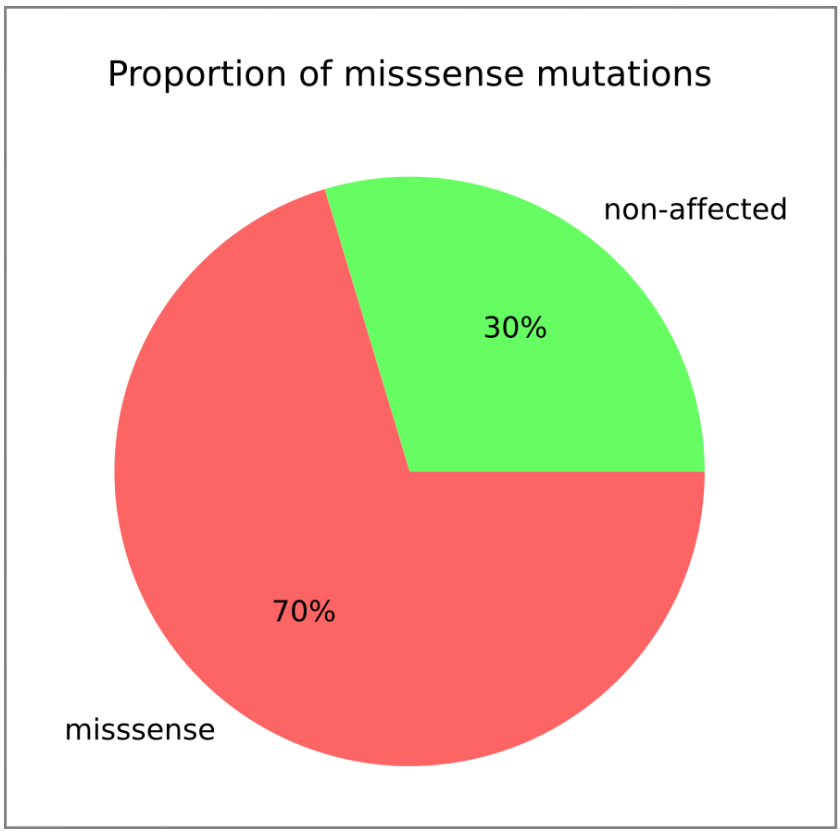

Fig. 1: Chart generated with average data from the 10 test cases, percentages are approximations

The other results analyzed were the proportions of different types of mutations. The 100% of the generated mutation, misssense or not, are divided into 4 categories. Methionine, stop signal, size and amino acid mutations. Methionine mutations are changes of the first codon which block the DNA-polymerase to start the read and transcription. The stop signal mutation is the mutation of the last codon, which will conduct to an incorrect read of the gene. Size mutations are the addition or deletion of a base to the snippet which shifts the

[2] The experiment was conducted with the previous provided DNA snippet of the OneNote page with a length of 12 bases, this could affect the final results.

complete gene one base and doesn't enable the formation of the original protein. And the amino acid mutation are those that produce changes on an amino acid in along the sequence.

The second part of the analysis consisted of looking to the proportions of the formation of different types of mutations. The more frequent are amino acid mutations (38%) because it is the largest snippet of the sequence where they are generated. Second, stop signal mutations (28%) followed by mutations in relation with the size of the string (22%). And last methionine mutations (12%). The percentages of the quantity of a certain type of mutation above the complete number of mutations can be seen in fig. 2.

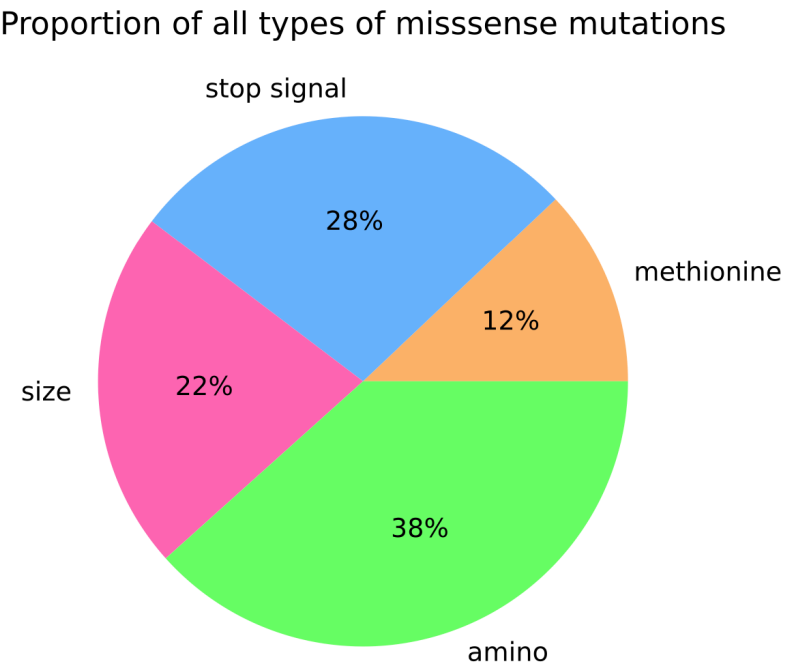

Fig. 2: shown data is the average on mutations above the total number of mutation generates per test case (1,000,000) percentages are approximations

Fig. 2 shows the percentages of different categories of mutations above the total number of misssense mutations. This means that the proteins that present changes are 38% due amino acids mutations, 28% due stop signals, 22% due the size of the sequence and 12% due the methionine encoding. As a conclusion it can be said, that the lower the number of affected cells, the lower the probabilities of having a mutation of that specific type. The percentages can be more or less accurately predicted studying the structure of the DNA, as the mutations are generated randomly. For example, knowing that methionine is only 1 codon of the sequence, it can be said that the probabilities of having a methionine mutation are the lowest, what can be checked by the experiment. According to the results of the simulation I would recommend to further study overall in amino acid and stops signal mutations, however we must research in all types of mutations shown here.

Conclusion

From this small simulation I can affirm that it shows clearly results that could be predicted with the theory. One could think that there should be some inconsistency in the code as the stop signal mutations appear to be stronger than the size mutations, as there are 3 different types of stop signal encodable. However this is subject of further study as the place where the size setback is generated could affect this result. All in all, I like to state that I have learned much more about python coding thanks to this project and I also expanded my knowledge and curiosity for biology and genetics. I think my research skill have also improved as I had to do research by myself and look up new information and I have also written this mini-article.

Justification of the results

In order to have accurate results and show there weren't any or minimal errors in the code 10 independent test cases were conducted as mentioned above. This 10 test cases provided 10 different numerical results, each of 1,000,000 mutations, that would compared in order to learn about the accuracy of the program.

Fig. 3 shows the evolution of the results along the 10 test cases. It can be seen that the difference between results minimal is and so do the evolutions of the categories of mutations separately. This means that all test cases draw the extremely similar results and that they are correct. If we have the maximum division (percentage) of two most different misssense mutations results as the value for the error margin, this is equal to 0.16%.

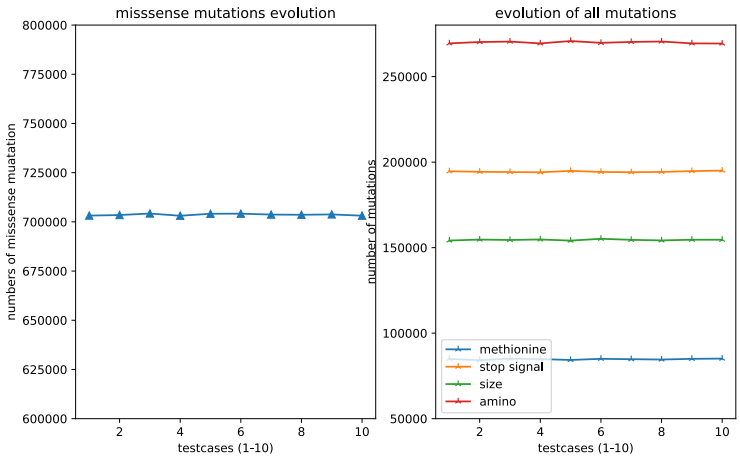

Fig. 3: left the evolution of misssense mutations along the 10 test cases, right the evolution of each category of mutation along the 10 test cases; all data is the original data from the simulation (no approximations)