Automatizing protein sequence and structure retrieval from UniProt

Experimental protocol design - Final report

Boulet Guillaume - Demore Loïc - Péreuilhet Iannis - Valentin Jules - Wawrzyniak Antoine

Phelma Biomedical Engineering - April / May 2024

Contents

Introduction

The development of computers has from the beginning been a huge asset for biology researchers; it allows indeed handling gigantic amounts of data while assisting them in their research process. More specifically, various tools were developed to offer solutions to precise issues : creating new formats to support lab-acquired data, setting up comprehensive databases to store it and make it available from anywhere, and building algorithms to sort and filter it, leading to major discoveries.

This paper will delve into the representation and use of proteins in bio-informatics. Since they make the bridge between DNA encryption and macro-scale effects on the cellular level, their study and understanding allows to tackle a significant part of the biology inquiries.

Objectives

The main objective of this project is to develop a set of tools to fulfill very specific demands for the purpose of research.

More precisely, the study of proximity and similarities between species and family of protein could allow to draw conclusions about the common behaviour of some of their proteins. This will be studied from a macroscopic point of view using taxonomy and trough a global database approach, but also from a sequence point of view using the BLAST algorithm and sequence alignement. These tools aims then to provide to the researcher an idea of which group or amino-acid is functionnal and why it is conserved trough the evolution, or not. Also, knowing where the amino acid is located in the structure permits to understand if it works with another group within the molecule; providing a proximity matrix will also be tackled.

To illustrate this project on a real research case, the choice of cement proteins (found in the barnacle Megabalanus rosa and the spider Nephila clavipes) known for their adhesive properties to diverse surfaces has been made [1]. Their study could lead to the making of bio-inspired adhesives with a large application field. Both Mrcp proteins from the barnacle and PySp proteins from the spider exhibit repetitive sequences in their structure, potentially facilitating self-assembly and adhesion through amyloid fiber formation.

1 A quick overview of Uniprot

UniProt is a comprehensive database containing manually annotated (Swiss-Prot) and computationally generated (TrEMBL) protein data [2]. It covers protein expression, localization, post-translational processing, disease associations, and references. The database integrates experimentally obtained and computationally predicted 3D structures. Searching is facilitated through keywords and an "Advanced search" engine, with tools like BLAST and ALIGN available for sequence comparison.

This tool is an amazing and almost exhaustive website to serve research in biology. However, since it's a public and free interface, some of the functionalities such as protein retrieving and sequence alignment are very time-consuming or not that handy when dealing with recursive research or when seeking to compare some aspects of proteins. After a few hours working on the website, we understood the necessity to create a client-side software which could dialog with the Uniprot service trough their API [3]. An API stands for *Application Programming Interface* and is a gateway provided by a web host which allows developers to make requests on their service and get a response. The uses of APIs will be detailed on the next section.

2 Data workflow in bio-informatics

This section will focus on explaining the process which was setup to retrieve data from Uniprot. This is the entrance point for all the analysis work that will be done afterwards.

2.1 Data Type

When we first dived into the files related to the world of proteins in bio-informatics, we were almost overwhelmed by the diversity of data types and storage methods. The summary of our research is presented in the table below :

	Fasta file	Features	PDB File	Image	Blast Result
Content	Lenght, AA Sequence	2nd Structure	3D structure	Specie's Image	Alignements
from	Uniprot	Uniprot	AlphaFold	Wikipedia	NCBI
Using	Rest API	Web API	Web API	Wiki web API	NCBIWWW
Format	.txt	.json	.pdb	.png	.xml

Table 1: Table summarizing the different files types used in this project

The wide variety of files types has encouraged us to be particularly rigorous on the data structure inside the software database. Main rules were to download the files only when necessary, and tag them as already present to not overwrite existing local data. This organisation has allowed to create a local mirror of the protein data files, such as :

- Fasta file; plain text containing the Uniprot ID (also called Entry), the specie and the common name of the sequence, and finally the complete amino-acid sequences;
- Protein data bank file (PDB) [4] ; mainly a list of 3D coordinates generated either by X-Ray spectroscopy or either by prediction using the Google Deepmind AlphaFold algorithm [5];
- All sort of renders created locally to be displayed for further analysis, more detail on the incoming sections;
- BLAST result files stored as XML and parsed using the Biopython NCBIXML module [6]

Setting up tools to ensure proper download and storage of these files was the first step to guarantee a straight-forward exploitation later. The flowchart on figure 11 sums up how data is handled in this project.

Since all of this takes place in the experimental protocol design framework, this report is more about following the path of the data all the way from Uniprot to the exploitable results we have created. This is not necessarily the chronological order.

The following section will then detail this automated process from a user point of view.

2.2 A convenient way to present proteins

We made the choice of simplicity concerning research from our tool called **Uniprot Retriever** : user can seek for a Uniprot ID he identified using UniProt research tool directly on the website (which is already exhaustive and very handy), or load it locally for previous search sessions. These two options are presented in figure 1.

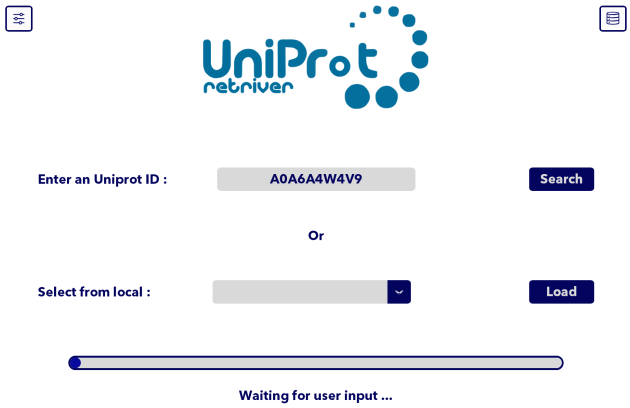


Figure 1: Screenshot from Uniprot Retriever starting frame

This being done, we had to find a design that let the user easily determine if he is in fact interested in this protein. To reach this goal, the following interfaces has been developped :

- The main protein ID card, called "Dashboard" (see fig 2) displays the main informations about the protein (locally-rendered 3D structure using Pymol [7], specie image retrieved from wikipedia [8]);
- A menu offering three possibilities, including our customized Blast tool, a 2D structure analysis and a whole taxonomy description, visible on the figure 3

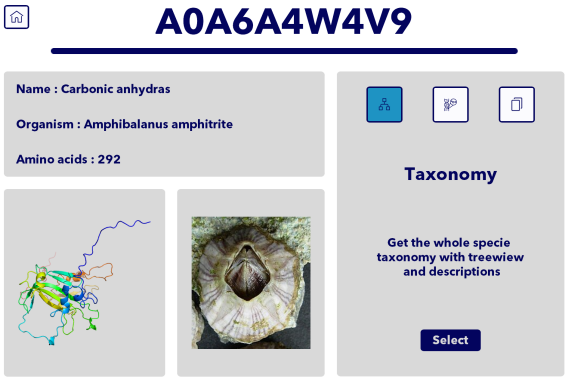


Figure 2: Screenshot of Uniprot Retriever main protein menu

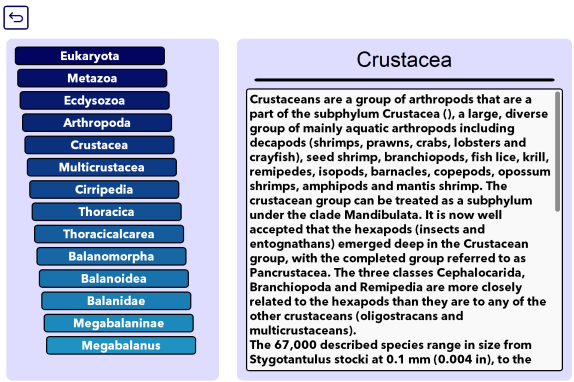


Figure 3: Screenshot of Uniprot Retriever menu to display protein taxonomy

This taxonomy tool is built using the specie name extracted from the Fasta file and the Wikipedia description of the concerned orders, ranks, etc. It aims to provide a quick idea of where the specie is located in the living world.

The last tool that has been setup before going in depth in the protein comparison tool is detailed in the next part of this document.

3 From 3D protein structures to 2D maps

The previous 3D render can be convenient for protein with simple to moderately complex structure, but for huge peptide chains it quickly starts to be bulky and sometimes unreadable. We have chosen to build a distance matrix to project this structure on a two-dimensional plane which will make it more easy to understand in some cases.

3.1 The process behind the two-dimensional renders

To reach this, an algorithm reads to protein positions from the PDB file and builds the inter-distance matrix by iterating on each amino-acid of the chain. You will find below a code snippet of this algorithm (4), as well as a render for a cement protein 20k (Q9GRC4) - 5

```python
def extract_CA_from_residue(residues):
    """ Extract the alpha carbon coordinates from a set of residues (aa) """
    CA = []
    for residue in residues:
        atoms = residue.get_atoms()
        for atom in atoms:
            if atom.get_name() == 'CA':
                CA.append(list(atom.get_coord()))
    return CA

def create_atom_matrix_CA(struct):
    """ Return a 2D matrix of the alpha carbon distances"""
    CA = extract_CA_from_residue(struct.get_residues())  # List of coordinates
    size = len(CA)
    matrix = np.zeros((size, size))
    for i, icarbon in enumerate(CA):
        for j, jcarbon in enumerate(CA):
            matrix[i, j] = np.linalg.norm(np.array(icarbon)-np.array(jcarbon))
    return matrix
```

Figure 4: Code snippet of algorithm building the 2D inter-distance matrix

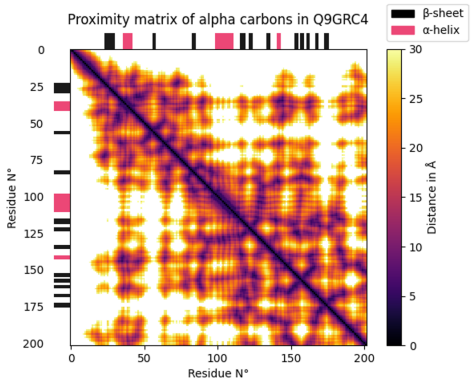


Figure 5: Screenshot of the visualisation of the inter-distance matrix

Note : For the purpose of reading, distances greater than 30 $\mathring{A}$ have been left blank. For optimization's sake, we choose to use only the position of alpha carbons in residues instead of working out the mass barycenter. Functions regarding these barycenter, hydrophobicity and charge are also developed but not integrated in the interface.

3.2 Comments and cross-validation

In order to assist the user in understanding the protein conformation, secondary structures are retrieved from Uniprot as inter-residues structures in a JSON file. This file is then read and its content is precisely displayed according to their position in the main peptide chain using black and pink rectangles directly on the figure.

To ensure this tool is properly working, let's cross-check it using our knowledge in bio-chemistry :

- The alpha-helix (in pink) are visible on the plot by highlighting a distance of 5 $\mathring{A}$ between an amino acid i and its neighbours $i-3$ and $i+4$, which is the radius of the helix;
- The presence of beta-sheets (in black) can be detected where $i-j>3$ (with adjacent sheet) with i and j two amino-acids indexes in the chain.

This part detailed how we developed tools in order to comfort the user before performing further analysis. We went all the way from raw to data to exploitable information and integrated this in a software to create a convenient workflow. Next chapter will focus on pushing protein alignments a step further.

4 Identifying similar proteins using BLAST

To meet the needs of researchers, we have set up a means of performing BLAST from within Uniprot Retriever. The purpose of the BLAST is to perform a comparative search between different amino acid sequences in order to find their closest neighbors in terms of alignment [9]. Online tools include the Uniprot and NCBI websites, both of which enable searches to be launched from a given FASTA file. In particular, these sites offer results containing up to a hundred similar proteins, with different comparison criteria such as percentage similarity, similarity score (calculated internally on NCBI) or e-value (value of the probability for which an amino acid sequence alignment occurs randomly in nature).

This last criteria is the one we'll be using in our subsequent searches. The main drawback of online BLAST queries is the **2-3 minutes** it takes to compute. Also, the search process can't be automated and is not very scalable, making the idea of integrating a local BLAST module into Uniprot Retriever all the more attractive.

The API furnished by the NCBI website initially enabled us to make BLAST requests to the online servers, but unfortunately each of these was also subject to a time limit of around 2-3 minutes. Similarly, when queries were sent in parallel, the user was restricted to a maximum of 10 search requests at a time, and was again subject to the servers busy schedules and response times. So the ideal solution we came up with was to run BLAST queries **locally** on a database previously downloaded by the user. The way the algorithm works is described in the diagram opposite 6.

- An initial query is carried out with a deliberately high e-value, in order to scan as many results as possible that are close to the initially chosen protein, as well as those that are further away
- Once all the proteins have been retrieved, we can process the results obtained locally to compare the appearance of new proteins for a variable e-value. This is particularly useful for segmenting different protein "families".

This algorithm saves a **considerable amount of time**, and eliminates the random aspect of the waiting time to which the user may be subjected. The only drawback concerns the local database, which can be downloaded from the NCBI website and can be quite large to contain on a personal machine. Initially, our tests were carried out on the SwissProt database, which proved to be sufficiently complete to guarantee a suitable data volume (about 35 GB $\approx 10^{10}$ amino acids).

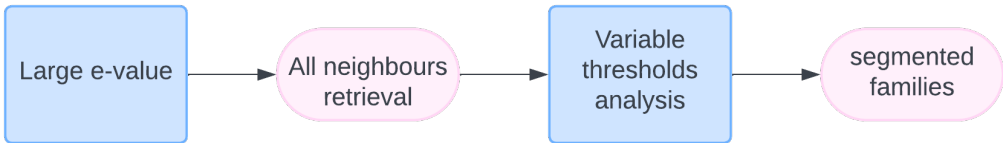


Figure 6: Flow chart of how the local BLAST algorithm works

The software opens the gate to other databases by making the access path customizable. A future installation on a dedicated LMGP computer is planned in order to run it on the exhaustive NR (Non-Redundant) database.

5 Pushing the research further thanks to Recursive blast

As indicated in the previous section, the effectiveness of BLAST search has been greatly enhanced by the calculation of the latter on a local basis. The time saved in this way leaves the door open to further automation of the process of searching for new protein families. In particular, an idea underlying the original BLAST is that other related proteins may exist but have not been found by a single BLAST calculation. The idea would be to use the n results obtained in BLAST step 1 to calculate a new BLAST search on each of the results obtained, in order to see whether new proteins not yet found might appear.

This algorithm, which is very costly in terms of queries and therefore computation time if carried out on NCBI or Uniprot websites, can be automated here with the local BLAST. Following this search, a classification is made between proteins already found and new proteins. The percentage of new proteins found on the basis of this classification is calculated for each new BLAST iteration. This percentage takes into account the number of occurrences of each protein, to account for the redundant aspect of the algorithm. The user can then take this percentage into account to determine whether or not it is useful to carry out a MULTI-BLAST search. The operation of the MULTI-BLAST algorithm is described below 7:

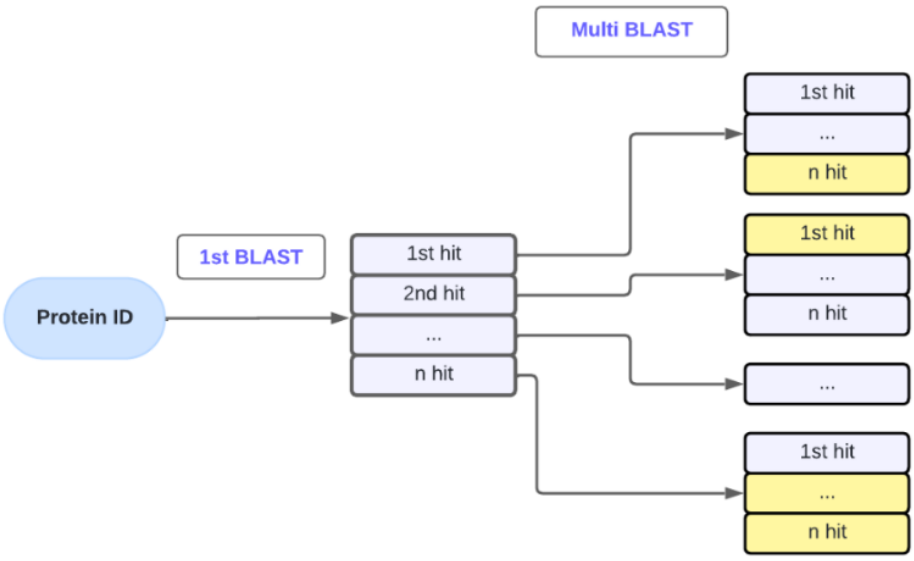


Figure 7: Workflow of MULTI-BLAST algorithm : the yellow color shows newly found proteins, while the blue color shows proteins already found

6 Alignement and amino-acids properties

Once the MULTI-BLAST were working, we searched to compare 2 protein families. The algorithm we developed compares the families according to their size, common amino acid sequences and location. It then displays the sequences according to the best alignment between the common parts, and highlights (in red) the common parts (see fig 8).

Biopython's alignment facility already exists and manage to do a similar work, and adds spaces symbolized by dashes '-' between the amino acids, in order to find the maximum number of amino acids in common. However, we wanted our own tool to display aligned sequences because the sequence of the reference protein is modified twice: once to match protein i and a second time to match protein j. So alignment of the 3 sequences is not possible because the reference sequence is not fixed, and using sequence copies does not work either.

The algorithm randomly selects a protein from each of the 2 families, and therefore 1 random sequence from each family. Each protein travels the reference protein so that each amino acid meets its peer on the reference protein :

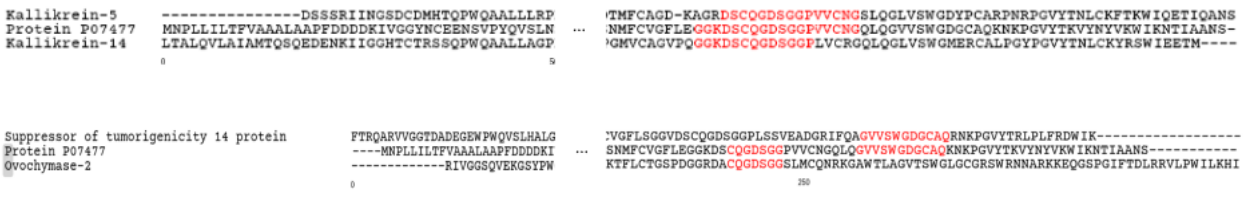


Figure 8: On top : alignment with 2 proteins from the same family. Below: alignment with 2 proteins from different families

The results of this protocol are as expected. By choosing to compare 2 proteins of the same family (which are apparently similar) we obtain an alignment of many amino acids side by side and located in the same

place compared to the reference protein. By choosing to compare 2 different family proteins we obtain an alignment of few amino acids and which are not located in the same place compared to the reference protein.

To go further we could have used the multi-alignment function of Biopython leaving the dashes added in order to **guarantee the spatial properties** of the proteins while fixing the reference protein. Indeed our protocol considers proteins to be linear without taking into account the spaces between amino acids.

7 Results and analysis

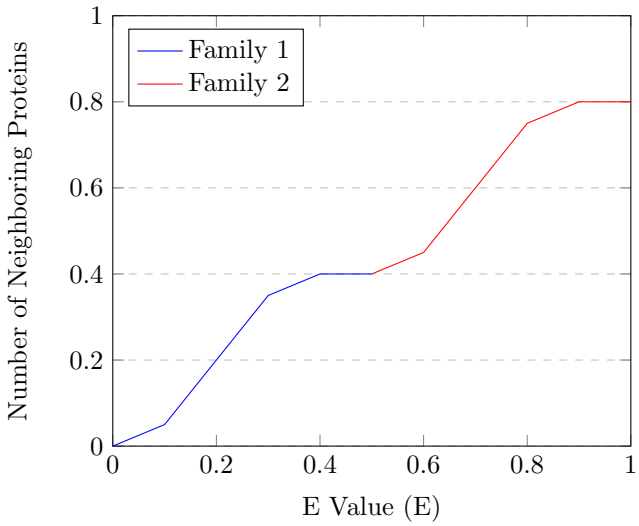
This part will explain and analyse the results obtained from the BLAST and MULTI-BLAST operations. Let's first recall that the goal is to find proteins similar to a protein of interest. It has been decided to classify hits into families of proteins. The sequences of proteins that belong to one family are similar.

7.1 Family Graph (BLAST)

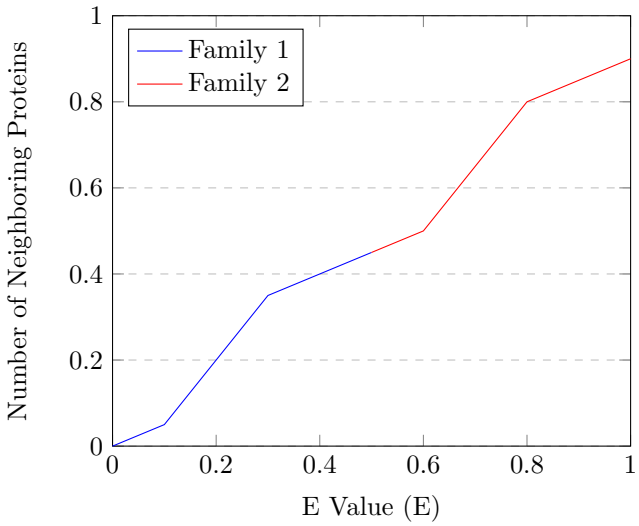
The the number of neighboring proteins (similar proteins) has been plotted as a function of the e-value (E). What we expect in a simple case is that for $E = 0$, the numbers of neighbors is 1 (the protein itself). Then when E increases, the number of neighbors eventually reaches a value m, at which there is a plateau. This corresponds to a family of m proteins. At the end of the plateau, when E increases even more, the number of neighbors eventually reaches a value n, at which there is a plateau, and so on. Our expectations in this simple case is displayed below.

In this simple case, finding each families would be equivalent to find each plateau, or when the derivative is 0.

In a more complex case, there is not any plateau, but rather a slighter increase of the number of neighboring proteins. Finding families would there be equivalent to finding the second derivative local maxima, or the inflexion points (up) of the curve.



Normalized graph of simple case family



Normalized graph on more complex family

This gives us a method to find families of proteins from a set of similar proteins from a BLAST. Here is an example of graph, which displays the families of proteins found from a BLAST on a carbonic anhydrase (Figure 9).

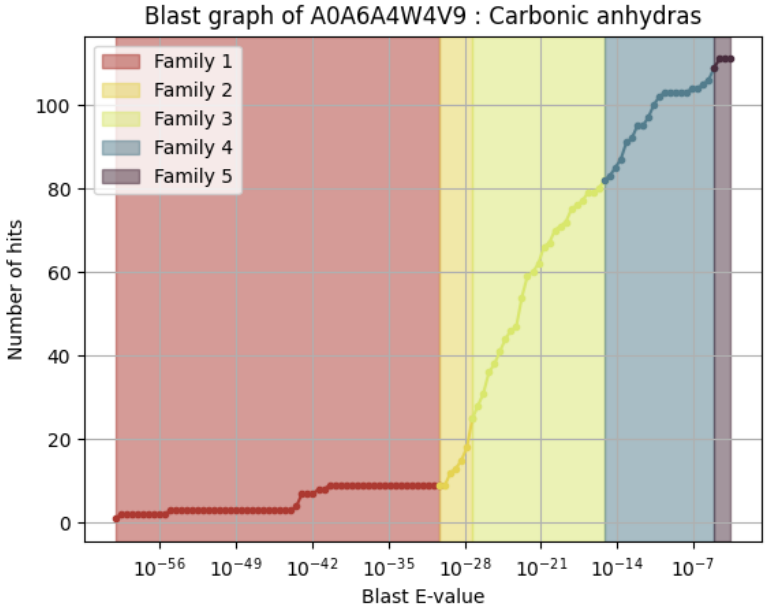


Figure 9: Example of BLAST graph (A0A6A4W4V9 protein)

7.2 Graph families (MULTI-BLAST)

This part details how the analysis of the result of the BLAST has been generalized to the MULTI-BLAST. The principle behind is to compute each BLAST graph. For the reading, only the mean and the standard deviation of the curves has been displayed, as well as the inflexion points, computed on the mean curve. Here is an example of MULTI-BLAST graph (Figure 10)

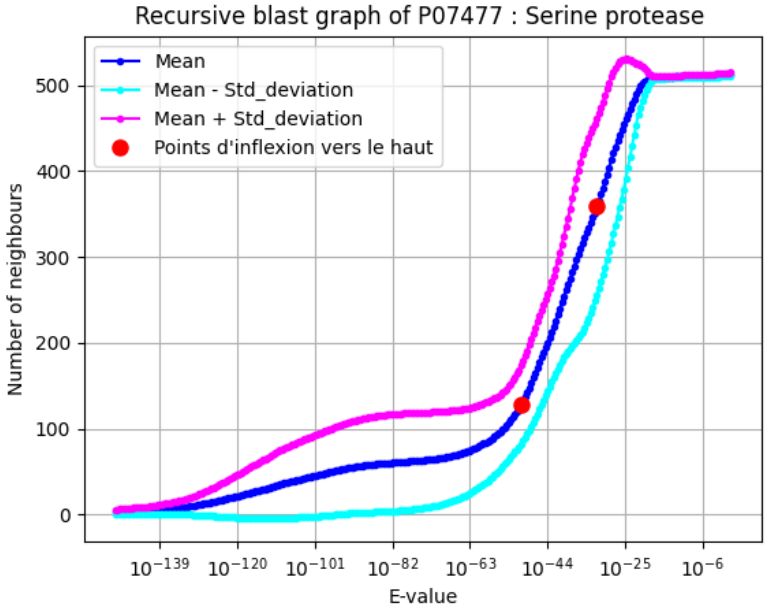


Figure 10: Example of MULTI-BLAST graph (P07477 protein)

The decision to plot the mean and its standard deviation is questionable. As for now, this is the best solution that we found. We first tried to plot the first curves found, which was not the best option in order to have an idea of the different families. With extra time, we could have tried to plot a random sample of all the curves, to have an idea of all the possible trajectories, since a lot of the trajectories are similar (on the examples we looked into). The choice of visualization for the analysis still need to be thought, and that is a perspective of improvement.

Other discussions are necessary about the inflexion points. The function that search them has some input parameters, that are fixed in this software version, and their value is the best compromise that we found. A possible improvement of the software would be to set these parameters to initial values, and to let the user change these values until he is satisfied with the inflexion points found.

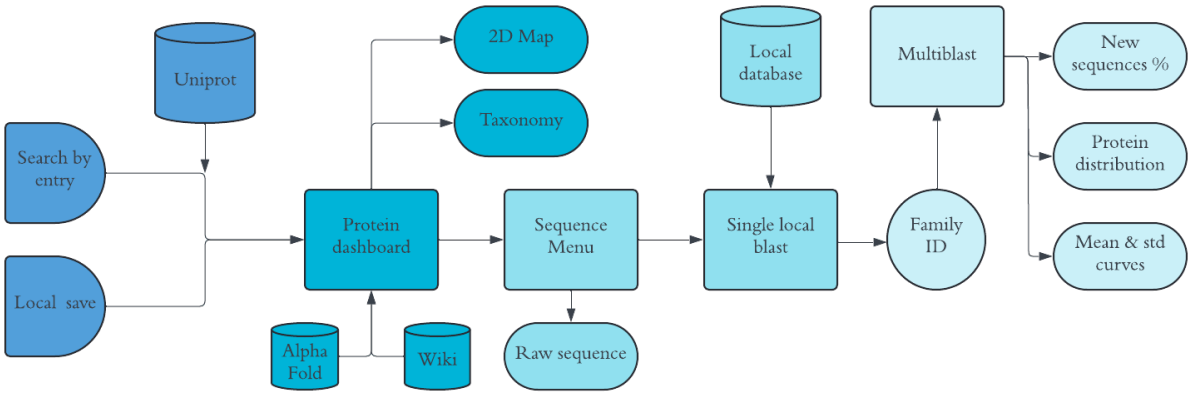


Figure 11: Flowchart summing up all the functionalities of the software

Conclusion and perspectives

During the development of this tool, we travelled all the way from zero level of understanding of databases and file formats, to the production of a complete and functional software package. Some aspects could not be explored due to lack of time and the diversity of the project, but many tools have been made available for future use. The advantages of running Blast locally proved to be significant, enabling the creation of new graphs that would normally be impossible to produce online.

This project gave us an insight into the world of bio-informatics, while linking it with our knowledge of biology acquired during the year. It also required a particularly rigorous organisation in the division of tasks, as we were all working on the same code base but on different aspects. Several choices and concessions were made to make it sustainable and re-usable for further optimisations.
[1]

[1] Codebase and software download avaible by contacting guillaume.boulet@grenoble-inp.org

Bibliography

[1] Mathilde Arque. *Identification des bases moléculaires de colles naturelles d'arthropodes.* PhD thesis, LMGP, Laboratoire des matériaux et du Génie Physique - Grenoble, 2016.

[2] The uniprot consortium. Uniprot: the universal protein knowledgebase. *Nucleic Acids Research*, 2023.

[3] Uniprot API Documentation, https://www.ebi.ac.uk/proteins/api/doc/.

[4] The Regents of the University of California. Introduction to protein data bank format. 2002.

[5] John Jumper et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

[6] Biopython documentation and tools, https://biopython.org/docs/1.76/api/index.html.

[7] R. Bryn Fenwick. Pymol reference card. Technical report, PyMOL Organization, 2007-2009.

[8] Martin Mahjlis. *Wikipedia-API Documentation.* Wikipedia Foundation, 2023.

[9] Tom Madden. The blast sequence analysis tool. *The national library of medecine*, 202?