Reda Batista

Audrey Fortuné

Master 2 : Bio Informatique

Années : 2020-2021

Projet 6 : Protein Interactions Calculator implémentation

I/ Introduction

Dans le cadre du projet, l'implémentation du script vise à ré-implémenter un web serveur déjà existant nommé PIC (*Protein Interaction Calculator*) disponible sur le lien suivant (http://pic.mbu.iisc.ernet.in/index.html). Le web serveur propose une interface sous forme de site internet prenant en entrée un fichier au format .pdb (*Protein DataBase*), de la protéine de notre choix. Le web serveur nous donne en sortie les différents types d'interaction que l'on désire. Le résultat est donné sous forme de plusieurs tableaux correspondant aux différentes interactions reconnues par le programme. Ceci est donné avec diverses informations comme par exemple les distances d'interaction ou encore les angles des composants.
Le but ici du projet est d'écrire un programme en langage de programmation python, en s'approchant le plus fidèlement possible la version originale du site PIC.
La conception d'un tel programme passe par plusieurs étapes que nous allons détailler plus loin. La première étape étant de mettre en place un workflow de développement le plus efficace. Pour ce faire nous avons décidé d'utiliser la méthode de développement *Kaban.*

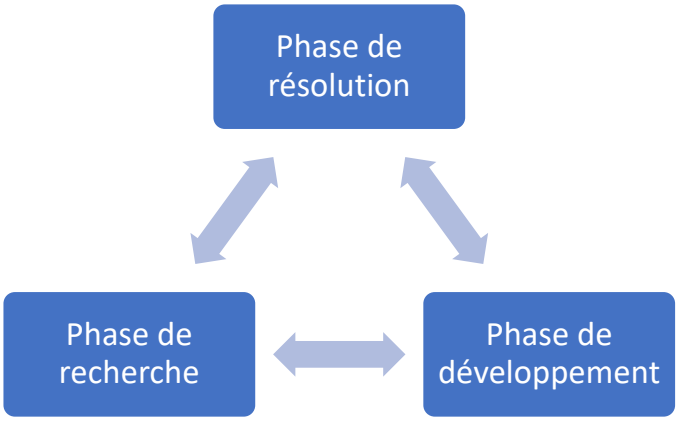
II/ Matériel et méthodes

1) Méthode agile

Pour définir au mieux nos besoins de production nous avons décidé d'utiliser la méthode *Kaban*. La construction du workflow en utilisant cette méthode nous offre plusieurs avantages qui correspondent à nos attentes de développement.
Pour remettre dans le contexte du projet, on peut séparer la phase de production en deux parties qui sont les suivants :

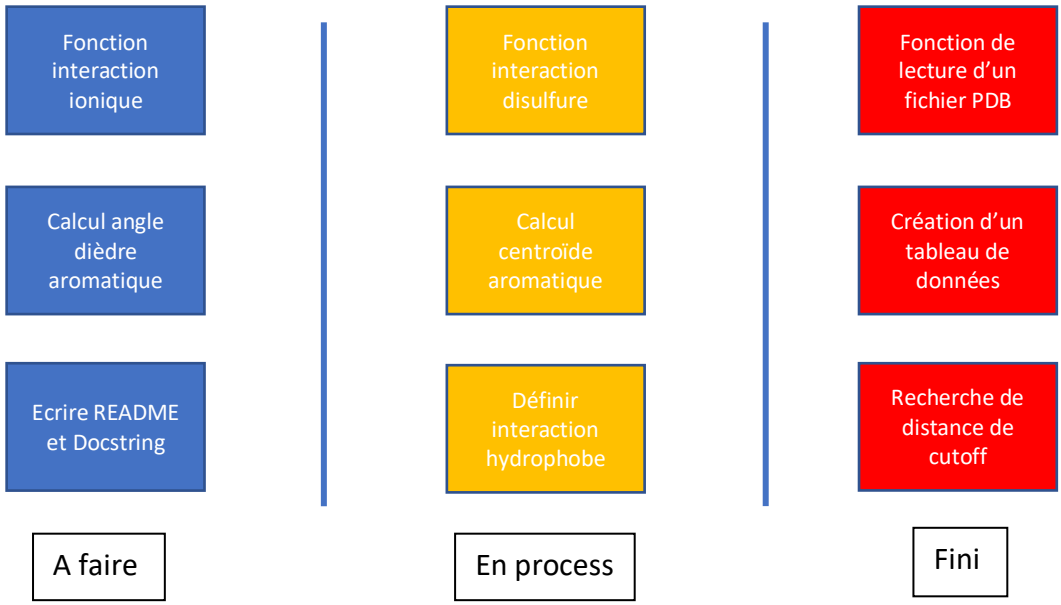
- la première qui s'apparente à de la recherche bibliographique, c'est-à-dire de la recherche d'information pour la détermination des interactions, des distances, des angles entre les différents atomes des acides aminés de la protéine ;
- la deuxième phase correspondant à la conception du programme proprement dit.

Nous avons dans la mesure du possible, essayé d'établir avant la mise en production du programme, les besoins nécessaires à sa réalisation en réunissant le maximum d'information. Cette organisation nous a posé problème. En effet, il a été compliqué de définir en amont l'ensemble des difficultés que la conception du programme allée nous poser. Il nous a donc fallu établir un seuil d'information pour l'initiation de la deuxième phase, puis au fur et à mesure d'établir de nouveaux seuils pour continuer. Nous avons défini donc trois phase de travail dans le diagramme suivant.



La première étape est la phase de recherche. Comme nous l'avons evoqué précedemment, il nous faut récuperer le maximum d'information pour initier le développement. La deuxième phase est la phase de développement correspondant à l'avancement du projet. La dernière phase de résolution est à une phase de transition. Cette étape définit le nouveau seuil d'information qui faut atteint en mettant à jour les nouveaux objectifs à atteindre.

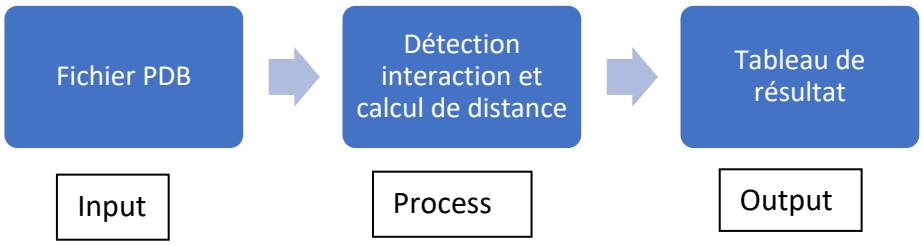
En suivant cette organisation, la phase clé de notre projet est celle de développement. Pour avancer au mieux sur cette phase nous avons utilisé un *Kaban board.*



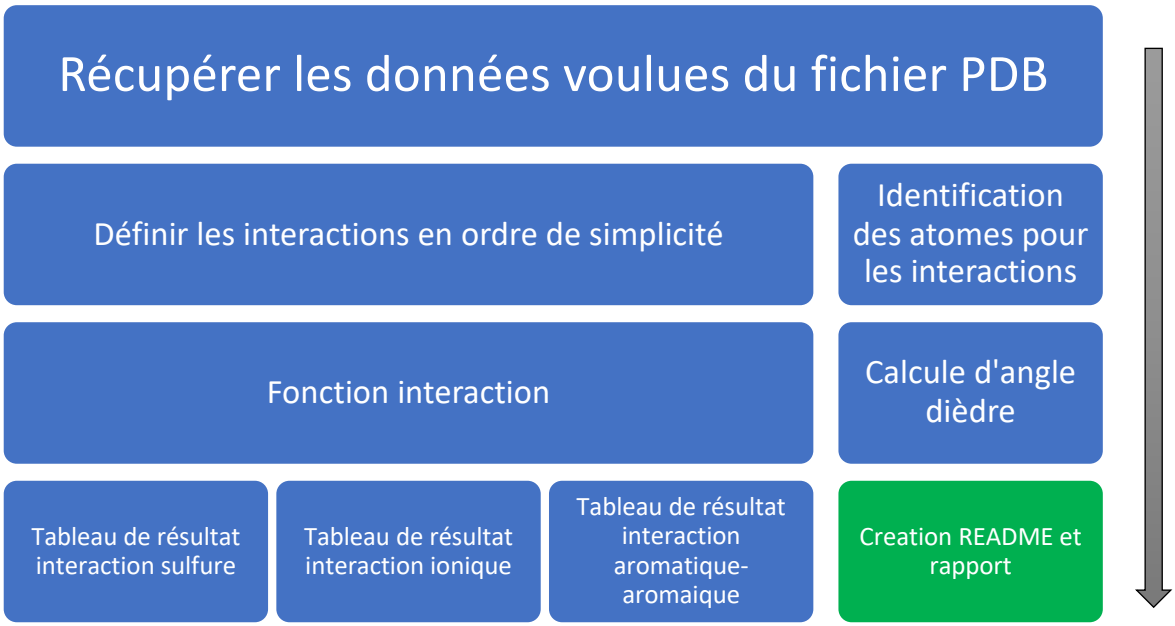
L'utilisation de planification comme sur l'exemple ci-dessus nous a permis d'avancer au fur et à mesure des contraintes que le code nous imposait. Pour expliquer comment nous avons pu développer ce programme, nous allons maintenant expliquer son fonctionnement ainsi que son implémentation.

2) Programme et implémentation

PIC est un programme qui reconnait les différentes interactions dans une protéine.
Le programme prend en entrée un fichier .pdb, et nous donne en sortie un ou des tableaux contenant l'ensemble des interactions du fichier. Pour tester notre script nous avons utilisé un fichier appelé « 2gkt.pdb ». Ce fichier est une inhibiteur d'hydrolase provenant du *Meleagris gallopavo* aussi nommé dindon sauvage. Notre choix s'est porté sur cette protéine pour plusieurs raisons : il nous fallait une protéine ayant une structure relativement simple pour tester basiquement notre code. Aussi il faut que cette même protéine contienne le maximum d'interaction pour avoir la possibilité de tester toutes les possibles interactions que nous avons implémentées.



Maintenant que nous avons introduit dans sa globalité PIC, nous allons expliquer sont implémentation.
Le schéma ci-dessous explique le dérouler de l'écriture du script.



L'implémentation de PIC se découpe en trois parties :

- L'extraction des données ;
- Le calcul des interactions ;
- L'affichage des résultats.

On peut supposer ici que la deuxième partie comporte une grosse proportion du programme. Dans notre script, l'extraction des données s'apparente à un *parser* qui va lire le fichier pour récupérer les lignes voulues. Cette fonction s'appelle *lecture* ().

En premier lieu, les fichiers .pdb ne contiennent pas les atomes d'hydrogènes. Nous avons donc dû utiliser un logiciel externe nommé « *molprobity ».*

Pour pouvoir travailler avec notre jeu de données, nous avons stocké sous forme de tableau appelé *table* l'ensemble des lignes extraites du fichiers (c'est-à-dire toutes les lignes commençant par « ATOM »).

Le script contient une fonction par interaction. Le principe de calcul d'une interaction est de repérer les atomes mit en jeu dans l'interaction, ensuite calculer leur distance grâce aux coordonnées et enfin comparer la distance à une valeur appelé *cutoff*. Ce dernier est une valeur (dans notre cas en angström) qui définit si une interaction entre deux atomes est possible. Par exemple pour les ponts disulfures, il est nécessaire que les atomes de soufres de chaque cystéine soient dans un rayon de 2.2 Ä pour interagir et donc former un point disulfure.

L'implémentation des fonctions dans leur globalité suivent cette même ligne directrice avec des points qui varient en fonction des interactions. Pour expliquer en détails l'implémentation des fonctions, nous allons prendre en exemple la fonction *ionic_interaction( )* qui permet le calcul des interactions ioniques. La fonction est divisable en plusieurs sous-parties que nous allons détailler.

```
i = 0
j = 0
compteur = 0
cutoff = 6
liste_distance = []
df_ionic_coo = pd.DataFrame((table.loc[((((table['AA'] == "ASP") & (table['atom'] == "OD1")) | \
                            (table['AA'] == "GLU") & (table['atom'] == "OE1"))]))
ionic_coo = np.array(df_ionic_coo)
df_ionic_nh = pd.DataFrame((table.loc[((((table['AA'] == "ARG") & (table['atom'] == "NH1")) | \
                        (table['AA'] == "LYS") & (table['atom'] == "NZ")) | ((table['AA'] == "HIS") & (table['atom'] == "ND1")))]))
ionic_nh = np.array(df_ionic_nh)
```

1

2

1 - Mise en place des différentes variables.

2 - Récupération dans notre dataframe *table,* des atomes des acides aminés nécessaires aux interactions ioniques, à savoir les résidus NH+ des résidus arginine, lysine et histidine, et des résidus COO- des résidus acide aspartique et acide glutamique. Nous séparons c'est résidus en deux tableaux (*array*) différents en fonction de leur charges.

```
for i in range(len(ionic_coo)):
        for j in range(len(ionic_nh)):
                distance = (np.sqrt(((ionic_coo[i,5] - ionic_nh[(j),5])**2) + ((ionic_coo[i,6] \
                                - ionic_nh[(j),6])**2) + ((ionic_coo[i,7] - ionic_nh[(j),7])**2)))
                liste_distance.append(ionic_coo[i,1])
                liste_distance.append(ionic_coo[i,0])
                liste_distance.append(ionic_coo[i,4])
                liste_distance.append(ionic_nh[j,1])
                liste_distance.append(ionic_nh[j,0])
                liste_distance.append(ionic_nh[j,4])
                liste_distance.append(distance)
                compteur += 1
```

3

4

3 – Calcul de l'ensemble des combinaisons de distances entres les interactions des charges positives et négatives. On itère sur les deux *array* en effectuant une double boucle.

4 - Cette étape permet de formater les résultats dans un tableau sous forme de liste que nous verrons plus tard

```
array = np.array(liste_distance)
array = array.reshape((compteur,7))
array = pd.DataFrame(columns = ['position', 'residue', 'chain', 'position', 'residue', \
                        'chain', 'distance'], data = array)
array['distance'] = array['distance'].astype(float)
array['distance'] = round(array['distance'], 2)
table_ionic_cutoff = array[array["distance"] < cutoff]
return(table_ionic_cutoff)
```

5

6

5 - Conversion de la liste ordonnée de l'étape 4 en *array* à deux dimensions, puis en *dataframe* pour l'affiche d'un tableau en sortie.

6 - Nous spécifions que la colonne des distances est un float arrondi à 10^{-2}. Puis nous sélectionnons les lignes dont les distances sont inférieures au *cutoff* prédéfinit dans les variables.

Après calcul, voici l'affichage dans un *shell*. Ce tableau n'est qu'une partie de l'affichage final.

```
Intraprotein Ionic Interactions
 Ionic Interactions within 6 Angstroms
     position residue chain position residue chain  distance
3          7     ASP     I       34     LYS     I      5.84
6         10     GLU     I       13     LYS     I      4.70
13        19     GLU     I       21     ARG     I      3.11
-------------------------------------------------------------
```

3) Les interactions

Les différentes interactions à implémenter dans notre code python, ainsi que les critères de sélection sont présentés dans le tableau suivant.

Interaction	Hydrophobes	Disulfure	Hydrogène	Ioniques	Aromatiques-Aromatiques	Aromatiques-Sulfure	Cation-Pi
Définition	Carbones chaines latérale	Deux soufres		NH+ et COO-	centroïdes	Centroïdes et Soufre	NH+ et centroïde
Résidus	ALA /VAL/LEU ILE/MET/PHE TRP/PRO/TY R	CYS	Tout les résidus	ARG/LYS/HIS/ASP/GLU	PHE/TRP/TYR	PHE/TRP/TYR/MET CYS	LYS/ARG/PHE/TYR/TRP
Cutoff	<5 Ä	2.2 Ä	4 A	<6 Ä	> 4.5 Ä <7 Ä	<5.3 Ä	6 Ä

III/ Résultats et discussion

La particularité du projet est qu'il nous a été possible de comparer les résultats obtenus directement avec les résultats donnés en sortie du serveur PIC. Dans la majorité de nos interactions, les résultats sont similaires avec ceux retrouvés sur le web serveur pour un même fichier .pdb. Les distances entre les (donneur – H – accepteur) sur les interactions entre deux chaines principales nous donnent des résultats différents car, l'ajout des hydrogènes ne s'est surement pas fait de la même manière que *PIC*.

L'une des différences entre les deux versions résident notamment dans le calcul des angles (par exemple avec l'interaction aromatique - aromatique). Avant de pouvoir écrire la fonction nous avons décidé de réaliser les calculs d'angles nous-même. Les résultats étant différents nous avons choisi de ne pas écrire la fonction. Nous avons néanmoins déduit que les différences de résultats peuvent être dû à des valeurs plus ou moins arrondie en fonction du support de calcul.

Parmi les fonctions à implémenter, celle pour les interactions hydrophobes n'a pas été calculé. La difficulté de compréhension et la manière d'implémenter la fonction ont joué dans notre décision de ne pas implémenter cette fonction immédiatement au détriment d'interaction plus simple à mettre en place.

Bibliographie

PIC : Protein Interaction Calculator ; **K.G.Tina and al.** ; Nucleic Acids Research ; 473-476 ; 2007

Comparative Protein Modelling by satisfaction of spatial restraints **A.Sali and T.Blundell** J.MOL.BIO ; 234,779-815 ; 1993

Conserved Spatially Interacting Motifs of ProteinSuperfamilies: Application to Fold Recognition andFunction Annotation of Genome Data ; **A.Bhaduri and al.** ; PROTEINS ; 657-670 ; 2004

A simple Method for displaying the hydropathic character of a protein **; J.Kyte and R.Doolittle** ; J.MOL.BIO ; 105-132 ; 1982

Interaction Geometry Involving Planar Groupsin Protein–Protein Interfaces **R.Saha and al**. ; PROTEINS ; 84-97 ; 2007

Aromatic-Aromatic Interaction: A Mechanism of Protein Structure Stabilization **; S. K. Burley and G. A. Petsko** ; Science ; 1982

A systematic analysis of atomic protein–ligandinteractions in the PDB ; **R.de Freitas and M. Schapira** ; MedChemComm ; 2017

PIC Protein Interaction Calculator : Criteria For Recognizing Various Types of interactions

Sulphur-aromatic interactions in proteins **; K.S.C. Reid and al** ; Elsevier ; Vol190-2 ; 1985

Inter-residue interactions in protein folding and stability ; **Gromiha MM, Selvaraj S.** ; Mol Biol. ; ;86:235–277. 2004

Comparative protein modelling by satisfaction of spatial restraints ; **Sali A, Blundell TL.** ; ;234:779–815. ; 1993

Interaction of DNA with clusters of amino acids in proteins. ; **Satyapriya R, Vishveshwara S ;** Nucleic Acids Res ;32:4109–4118. ; 2004