

Rapport de Projet UE Python

Sujet 3 : Abondance et stickiness

Table des Matières

Table des Matières	1
Introduction	2
Partie 1 : Analyse	3
Problématique	3
Travail effectué	3
Partie 2 : Prédiction	11
Problématique	11
Travail effectué	11
Discussion	12
Références	14

Introduction

Ce projet *Abondance et Stickiness* a pour objectif d'approfondir le concept de stickiness. Pour rappel, la stickiness est une valeur reflétant la propension pour chaque type d'acide aminé d'être impliqué dans des interfaces protéiques (spécifiques ou non) [1].

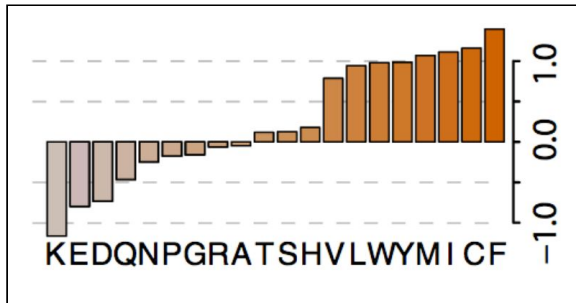


Figure 1 :
représentant l'échelle de stickiness
proposé par Levy et al., 2012 [1]

A l'aide de ce concept, il est possible d'étudier les notions et impacts liés aux interactions protéiques indirectes qui sont toujours très difficilement caractérisable expérimentalement.

En effet, on peut supposer qu'une protéine ayant une stickiness globale élevée en surface sera plus encline que d'autres à interagir de manière non-spécifique dans un contexte cellulaire. Sous cette hypothèse, une trop forte propension aux interactions non-spécifiques peut induire des effets délétères comme la formation d'agrégats protéiques.

Plus spécifiquement, dans ce projet sera approfondi l'étude de la corrélation entre stickiness globale de surface de protéines et l'abondance de ces mêmes protéines; ainsi qu'une tentative de prédiction des régions de surface à stickiness élevée.

Partie 1: Analyse

Problématique

Dans cette partie, on cherche à montrer la corrélation entre l'abondance d'une protéine au sein d'une cellule, et sa stickiness globale. Cela amène deux sous-problèmes, celui de la séparation des acides aminés de surface des acides aminés interne de la protéine, et celui de l'évaluation de la stickiness globale d'un ensemble d'acides aminés.

Travail effectué

Tout d'abord, il a fallu récupérer la liste des protéines pour lesquels une valeur d'abondance était disponible et un fichier pdb existant. Pour cela, on parse les différents fichiers, puis on interroge la base de données pour récupérer les pdb correspondants.

Cela nous a permis de récupérer 1075 fichiers pdb.

Une fois ce premier traitement effectué, il faut ensuite une méthode pour traiter un fichier pdb, extraire ses acides aminés de surface, et lui attribuer sa stickiness.

La méthode utilisée ici pour récupérer les acides aminés de surface est la méthode CV (comme développé au cours de l'UE).

La méthode CV est un moyen de déterminer l'enfouissement de l'acide aminé, en étudiant le nombre d'éléments autour de celui-ci.

Une première version manipulant 3 grands vecteurs correspondant respectivement aux coordonnées en x, y et z des atomes de la protéine est ainsi implémentée.

Cette fonction va, à partir d'une liste contenant les coordonnées des atomes d'un acide aminé (ou de son barycentre), itérer sur les coordonnées de chaque atome de la protéine et cela afin de déterminer les atomes à moins de r_c angstrom du résidu.

Mais de part la haute complexité du problème, il est nécessaire d'optimiser le calcul de la CV. Par conséquent une autre fonction manipulant la transposée du vecteur x, y, z de la première version est implémentée, celle-ci étant plus optimisée sur les accès mémoire.

Étant donnée que le calcul de la CV pour chaque fichier PDB est la partie la plus lourde en calcul de part le nombre important de fichiers à traiter, il est nécessaire de sauvegarder les résultats de CV obtenus pour chaque acide aminé d'une protéine

dans un fichier. Cela permet par la suite de ne pas avoir à recalculer la CV hormis dans le cas d'une variation de la distance rc utilisé pour son calcul.

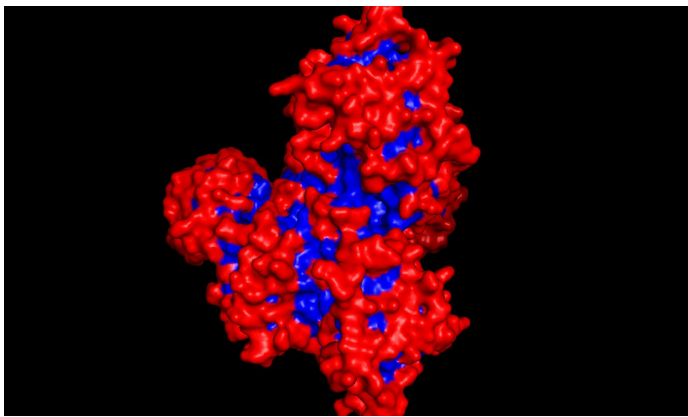
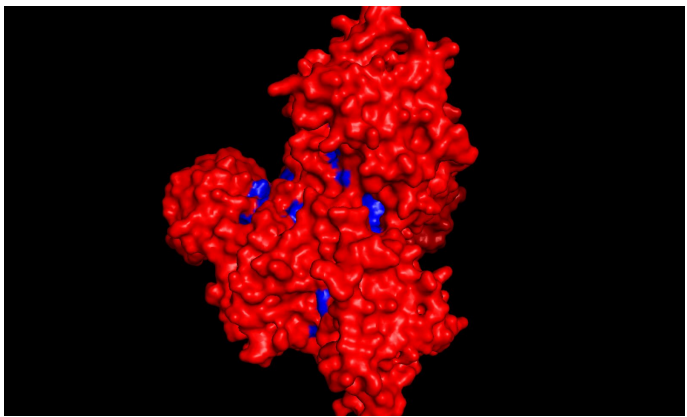
Pour cela, on sauvegarde pour chaque fichier PDB, un fichier .cv qui lui est associé, et qui contient les coordonnées de chaque résidu dans le pdb, et la valeur de CV du résidu. Cela permet de pouvoir extraire directement les informations calculées précédemment.

Toujours dans une optique de diminution du temps de calcul, on utilise les coordonnées du barycentre d'un résidu pour calculer sa CV plutôt que les coordonnées de chacun des atomes qui le composent.

Enfin il reste à déterminer un "seuil" pour la CV afin de déterminer si le résidu est considéré en surface ou non.

Pour cela on écrit dans le b-factor des pdb une valeur booléenne : 0 pour un résidu enfoui, ou 1 pour un résidu exposé.

Nous obtenons les résultats suivants :

<p>Figure 2 : Représentation de la sélection de la surface avec les paramètres : rc=10, seuil=0.75</p>	<p>Figure 3 : Représentation de la sélection de la surface avec les paramètres : rc=20, seuil =0.75</p>
	

En rouge, les résidus considérés comme étant à la surface.

On remarque qu'en utilisant une distance rc de 10 pour le calcul de la CV et un seuil de 0.75, nous obtenons une séparation des résidus de surface très satisfaisante. Nous avons donc utilisé ces paramètres la pour la suite du projet.

Maintenant que les résidus de surfaces ont été extraits, il faut maintenant calculer la stickiness de chaque protéine.

La stickiness d'un groupe de résidus se calcule en sommant la stickiness de chaque résidu. Par conséquent, trois groupes de résidus peuvent être distingué pour chaque protéine : l'un correspondant aux résidus de surface déterminés par la méthode CV;

l'autre correspondant aux résidu enfouis; et un dernier correspondant à la globalité des résidus.

Enfin on calcule la stickiness pour chaque groupe en sommant la stickiness de chacuns des résidus composant le groupe de résidus.

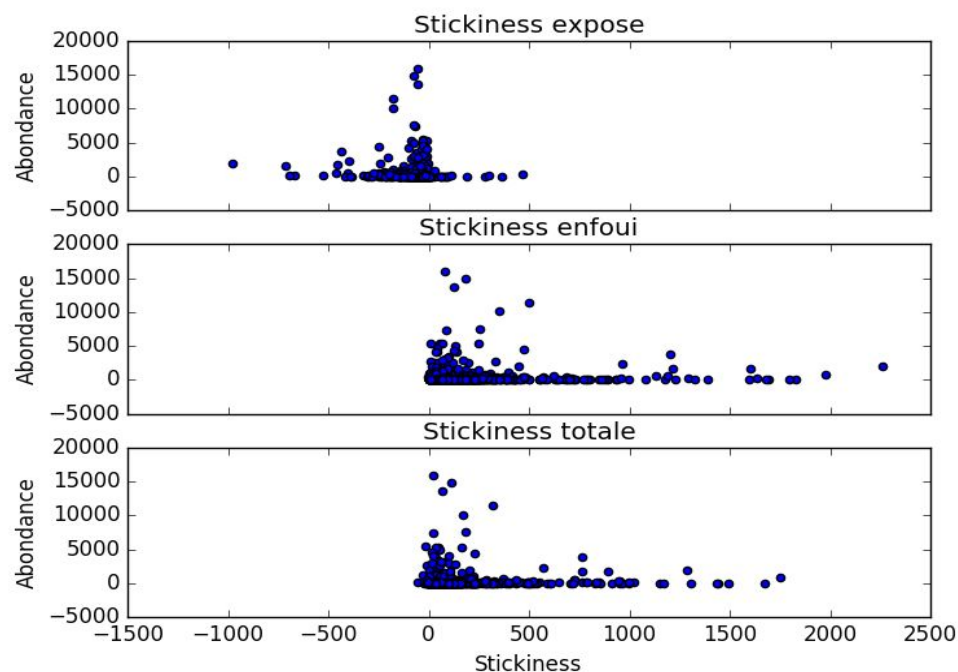
On ajoute une option de normalisation, en divisant le résultat obtenu par le nombre de résidu comportant le groupe afin de comparer les résultats entre eux.

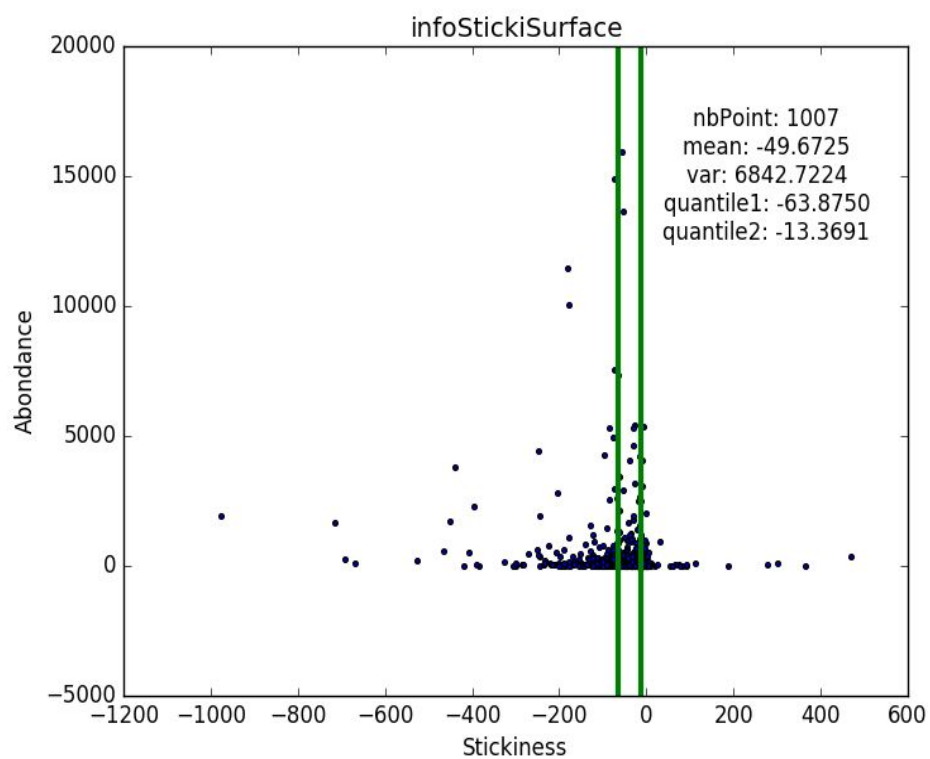
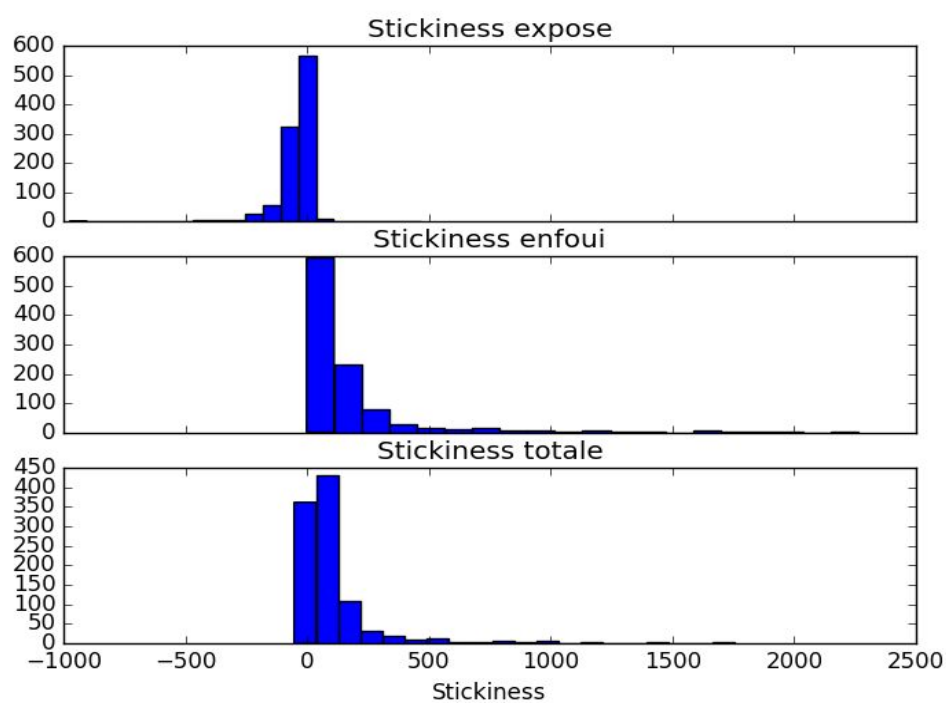
Finalement, on associe à la protéine son abondance et sa stickiness afin de pouvoir étudier graphiquement les résultats.

On obtient ainsi les résultats suivant:

Figure 4 :

Graphiques des résultats obtenues avec : Distance = 10, seuil = 0.75, méthode = CVbarycentre, sans normalisation





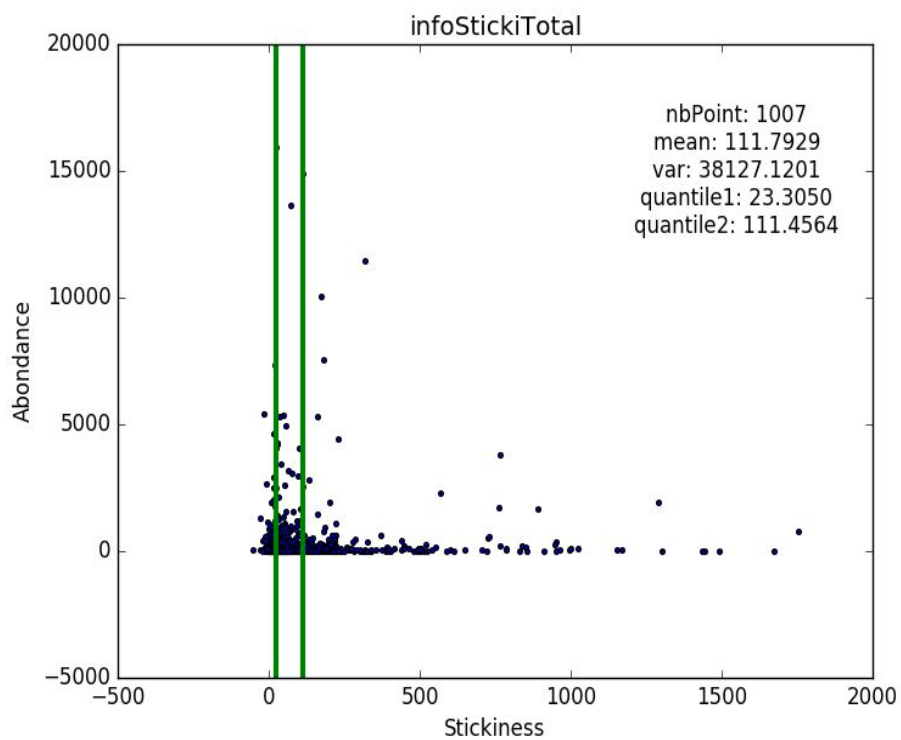
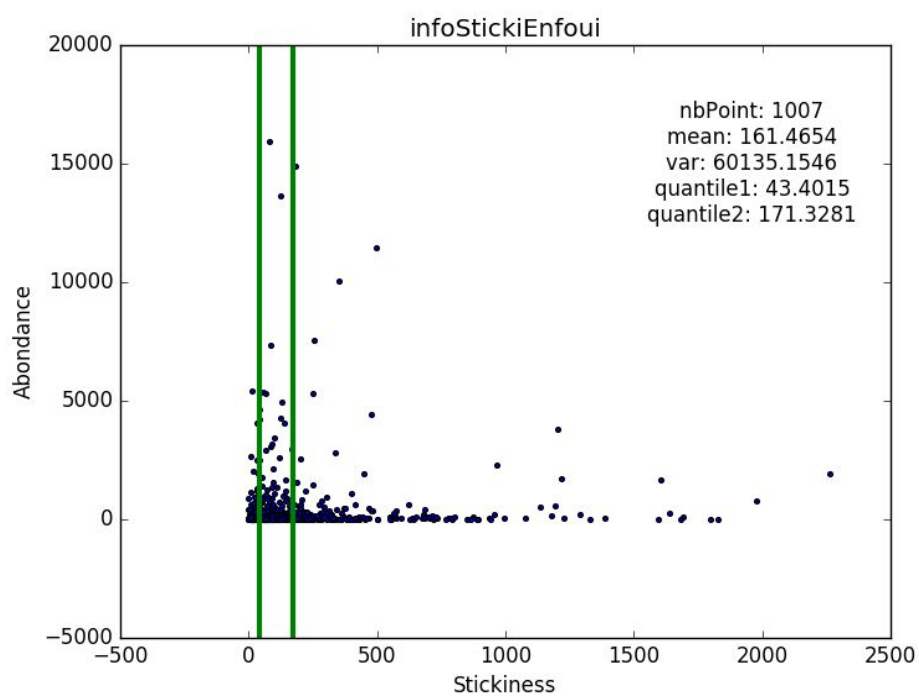
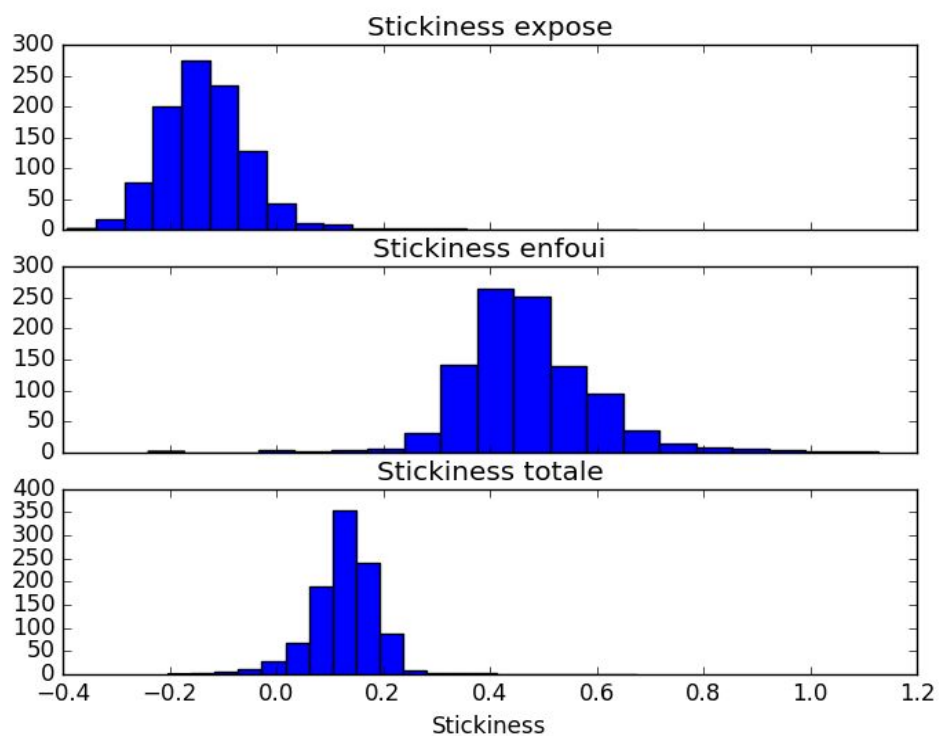
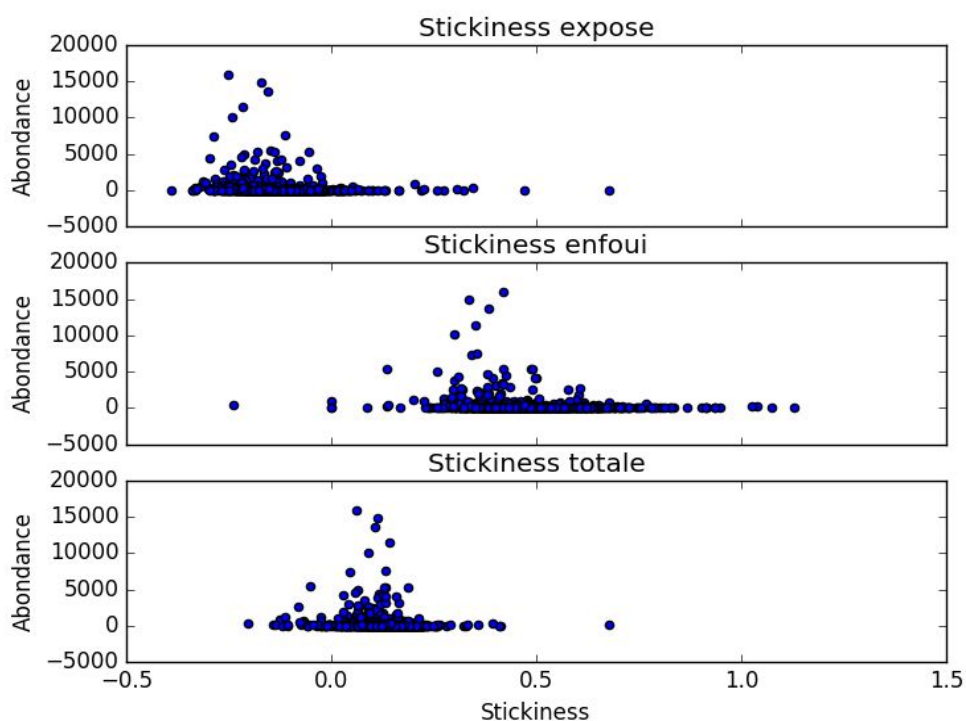
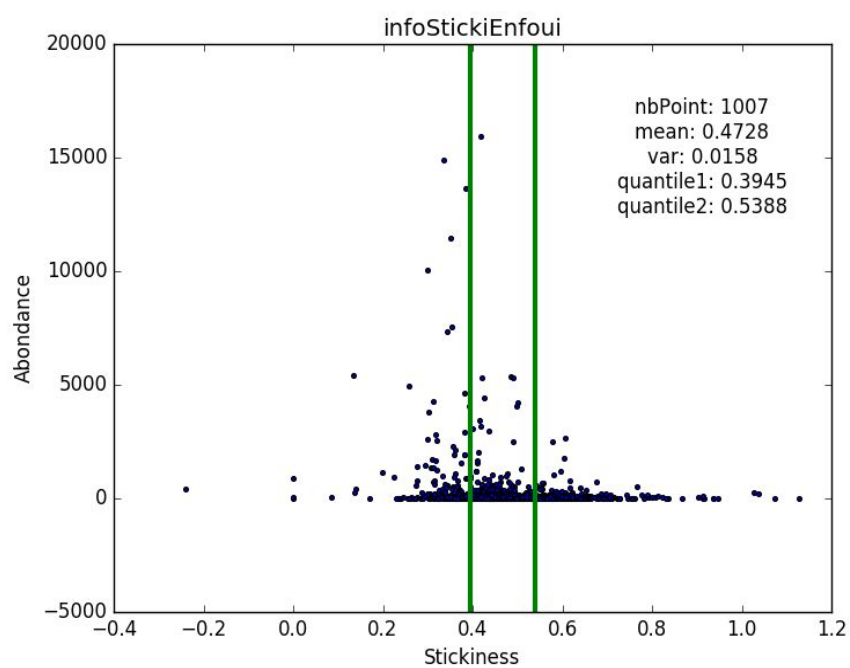
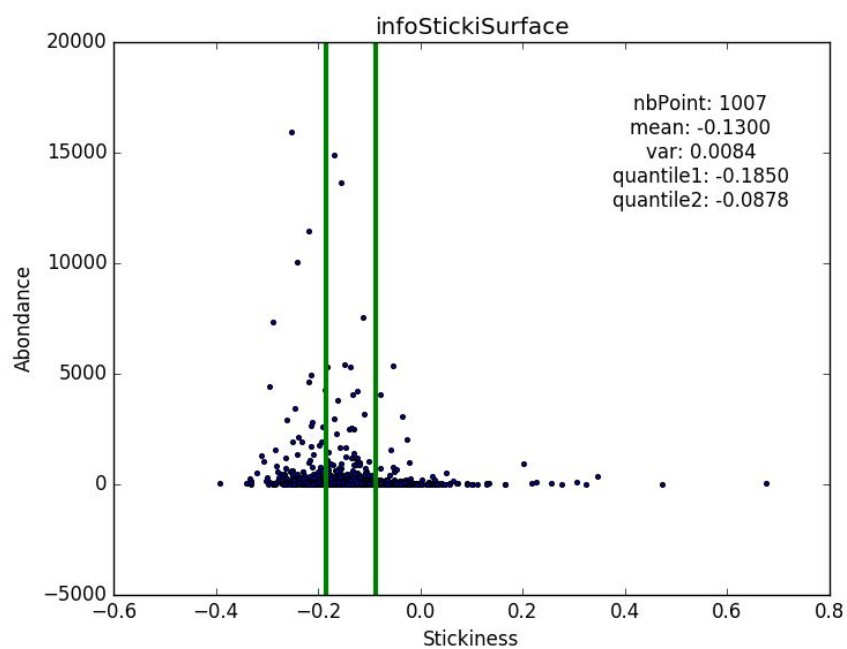
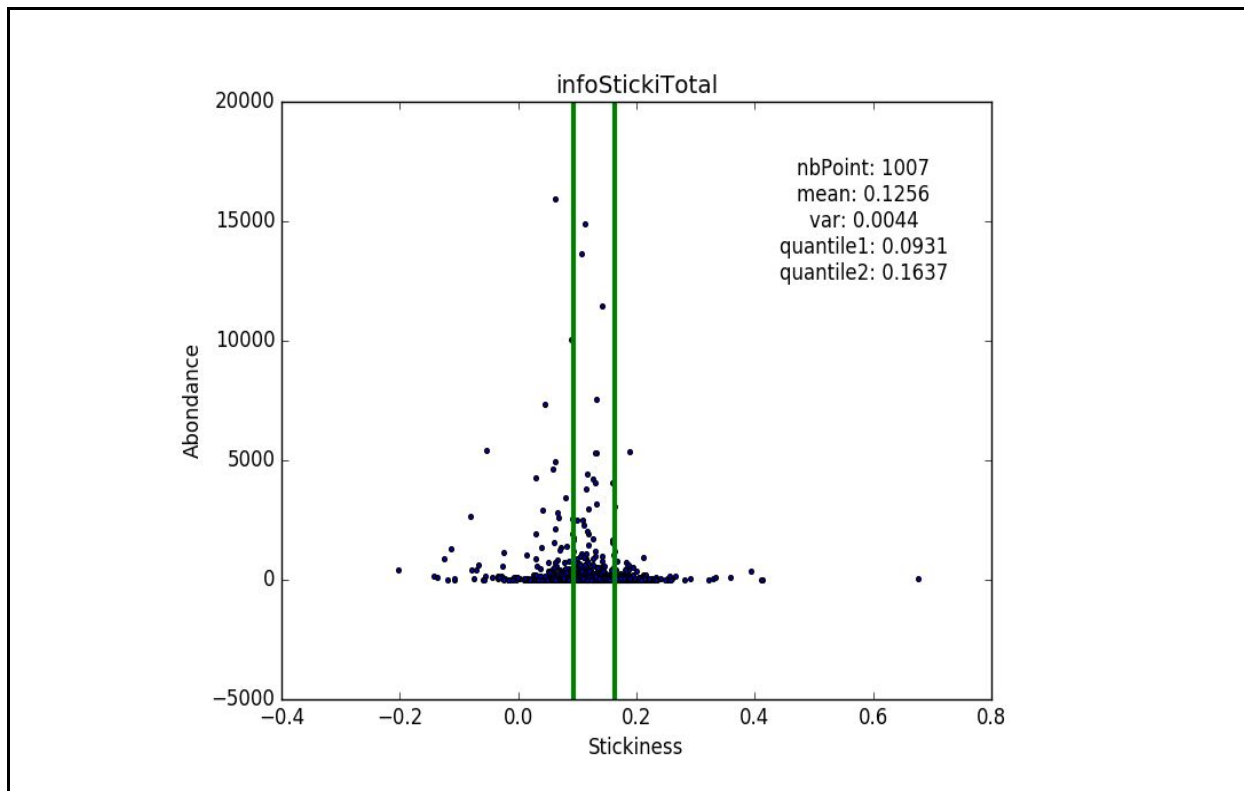


Figure 5:

Graphiques des résultats obtenues avec : Distance = 10, seuil = 0.75, méthode = CVbarycentre, normalisation par nombre de résidu







On remarque tout d'abord sur les graphiques non-normalisés une nette différence entre le coeur et la surface de la protéine.

La surface est clairement non-sticky ce qui est tout le contraire du coeur. C'est d'autant plus le cas que les quantiles (lignes horizontales vertes) n'ont jamais la valeur 0 comprise entre leurs bornes. Cela soutient l'hypothèse d'une sélection positive des protéines à faible stickiness de surface car l'inverse entraînerait des effets délétères.

En observant les graphiques normalisés, on remarque que les points semblent évoluer selon une loi gaussienne. Cela suggère qu'il existerait une valeur de stickiness optimale favorisant l'abondance de la protéine dans la cellule. Cette valeur serait de -0.13 pour la stickiness de surface ou 0.125 pour la stickiness totale.

Partie 2 : Prédiction

Problématique

Dans cette partie, on cherche à prédire le(s) site(s) d'interaction d'une protéine de structure connue. Pour ce faire on utilise le critère de stickiness sous l'hypothèse que les régions enrichies en résidus à haute stickiness sont des indicateurs d'une plus forte propension à participer à des interactions.

On cherche donc à prédire des zones en surface enrichies en résidus à haute stickiness.

Travail effectué

Tout d'abord, les structures de protéines connues ont été extraites de fichier PDB car facile à manipuler et pratique pour la visualisation des résultats grâce au logiciel PyMol et au champ 'bfactor' modifiable pour chaque atome.

La première étape consiste en la séparation des résidus de surface par rapport aux résidus de coeur de la protéine. En effet, on cherche à prédire des zones d'interaction donc situées en surface par définition. Pour faire cela, la méthode de calcul de CV avec un seuil rc de 10 Angstrom, couplé à un seuil de CV 0.75, a été utilisée et permet une séparation très satisfaisante comme vu précédemment.

Habituellement, l'ensemble des informations du fichier PDB est stocké sous la forme de plusieurs dictionnaires imbriqués. Ici pour plus de facilité algorithmique et une optimisation du temps de d'exécution, un numpy.array (cf le package numpy) stocke plusieurs informations (lignes) pour chaque résidus déterminés comme en surface (colonnes). Ainsi ce numpy.array présente 7 lignes :

1. le nom de la chaîne à laquelle appartient le résidu
2. l'indice du résidu dans le dictionnaire PDB
3. la coordonnée en x du résidu
4. la coordonnée en y du résidu
5. la coordonnée en z du résidu
6. la valeur de stickiness du résidu
7. le numéro de cluster du résidu

A cette étape, les informations nécessaires pour remplir les cinq premières lignes sont récupérées tandis que les deux dernières sont initialisées à 0 et seront remplies plus tard dans le programme.

NB : le `numpy.array` présente donc autant de colonnes que de nombre de résidus déterminés comme en surface grâce à la méthode CV.

Ensuite, la valeur de stickiness est affectée à chaque résidu de surface (colonne 6 du `numpy.array`). Pour ce faire une table contenant une valeur de stickiness pour chaque type d'acide aminé est utilisée.

NB : Une version du programme écrit directement cette valeur de stickiness (normalisée sur un intervalle voulu) dans le champ 'bfactor' afin de pouvoir visualiser sous PyMol la répartition des différentes valeurs de stickiness sur les résidus de surface de la protéine. Cette visualisation servira pour vérifier les régions prédites par la suite (figure 7).

Un nouveau tri des résidus à considérer pour la suite du programme est alors réalisé. Les résidus dont la stickiness est inférieure à un certain seuil (empiriquement 0) ne sont pas pris en compte pour la suite du programme et leurs colonnes respectives sont supprimées du `numpy.array`.

Il s'agit ensuite de réaliser un clustering des résidus proches à stickiness élevée.

Pour cela, le programme cherche en premier lieu le centre d'un nouveau cluster en prenant le résidu présentant la plus haute stickiness parmi ceux qui ne sont pas encore intégrés à un cluster.

Une fois le centre déterminé, les résidus proches de ce centre (ayant forcément un stickiness supérieure au seuil significatif) sont ajoutés au cluster selon deux limites : une limite de distance entre le barycentre du résidu central et les barycentres des résidus intégrés ainsi qu'une limite du nombre de résidus par cluster (seuil empiriquement évalué à 30 résidus au vu des connaissances des interactions protéiques).

Le numéro du cluster qui vient d'être rempli est alors affecté à chaque résidu le composant (colonne 7 du `numpy.array`).

Cette construction de cluster est répétée autant de fois que le nombre voulu de cluster pour la protéine.

Le numéro du cluster est enfin écrit dans le champ 'bfactor' de chaque atome composant respectivement chaque résidu du `numpy.array` afin d'être visualisé (les autres ayant déjà été initialisés à 0).

Discussion

Le programme ainsi réalisé a pour avantage une rapidité d'exécution très élevée de quelques secondes même pour des protéines de taille conséquente.

Pour présenter les résultats du programme, le fichier '1AC5.pdb' est utilisé car il est de taille conséquente et présente ainsi une grande surface.

Sur les figures suivantes extraites de PyMol, l'image de gauche représente la prédiction de 15 clusters et celle de droite simplement la stickiness (les régions blanches étant les moins sticky et les rouges les plus sticky).

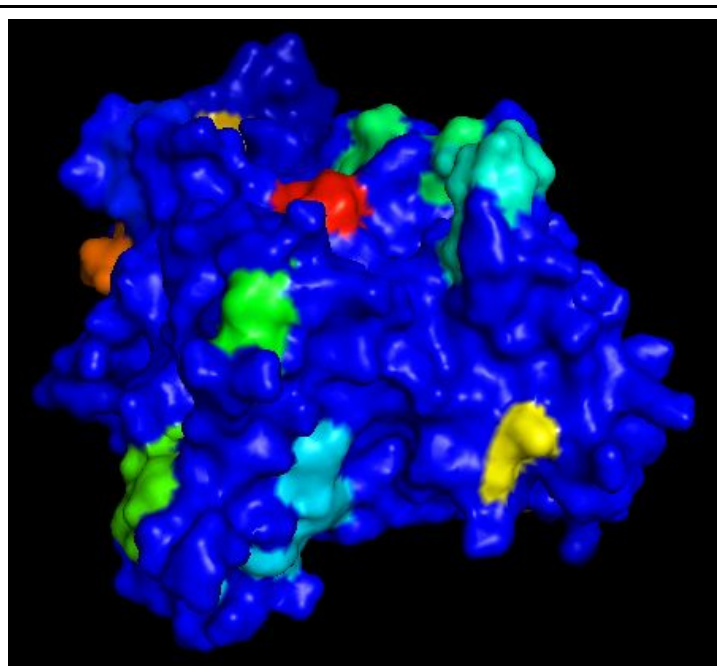


Figure 6 :

Représentation de la prédiction de 10 clusters sur le fichier pdb 1AC5

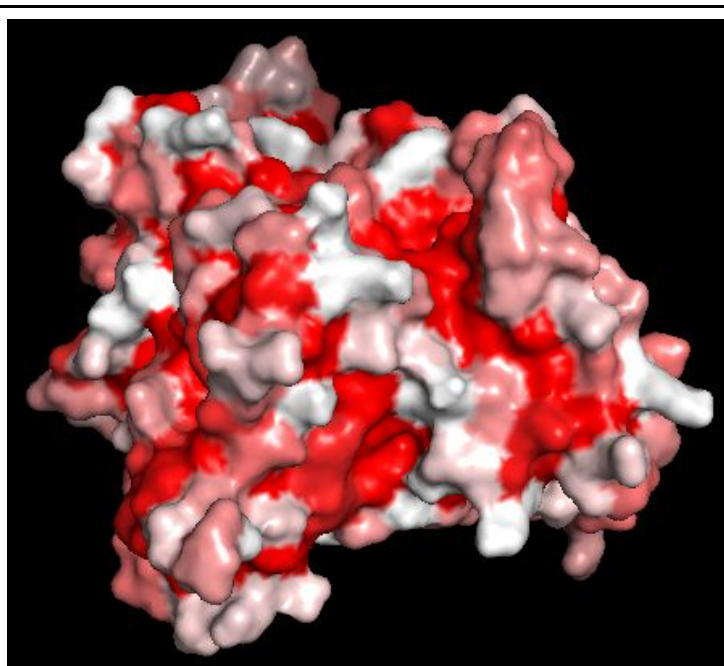


Figure 7 :

Représentation du mapping de la stickiness des résidus de surface sur le fichier pdb 1AC5

On observe que les zones prédites sont bien des zones présentant des résidus à stickiness élevée. En cela le résultat est concluant et on peut affirmer que la prédiction est correcte (la prédiction a été vérifiée sur plusieurs autres protéines).

Il faudrait en plus de cela vérifier si les régions prédites sont effectivement des régions d'interaction connues et démontrées expérimentalement.

On remarque cependant que certaines régions intermédiaires en stickiness sont prédites alors que d'autres régions clairement plus sticky ne sont pas retenues. Cela s'explique simplement par l'absence de prise en compte de la stickiness lors de la construction du cluster après le choix du centre effectué, comme détaillé par la suite.

Un défaut très clair du programme est qu'une fois le résidu de plus haute stickiness défini comme centre, le cluster se construit uniquement en fonction de la distance au centre. En effet si on ajoute au cluster les résidus les plus proches du centre, on ne maximise pas forcément la stickiness du cluster ainsi formé. Deux solutions à ce défaut pourraient être approfondies :

- la mise en place d'un score prenant en compte la distance mais aussi la stickiness pour choisir les résidus à associer au cluster.
- on pourrait très simplement garder le fonctionnement d'ajout de résidus uniquement en fonction de la distance mais construire un cluster pour chaque résidu de surface et calculer la stickiness totale puis ne retenir que les n meilleurs (parmis tous les clusters possibles). Cette solution impacterait cependant de manière non-négligeable le temps d'exécution.

Références

[1] Levy ED, De S, Teichmann SA. Cellular crowding imposes global constraints on the chemistry and evolution of proteomes. *Proc Natl Acad Sci USA* 2012;109:20461-6. doi:10.1073/pnas.1209312109.