Projet ContactMap

Contexte :

Les cartes de contact (contact maps : https://en.wikipedia.org/wiki/Protein_contact_map) sont un moyen commode pour représenter la structure d'une protéine sous forme visuelle, avec moins de complexité que la structure 3D complète. Cela peut servir par exemple à aligner la structure de protéines homologues, ou tout simplement visualiser sous forme d'image une sorte « d'emprunte » de la protéine. On se propose ici de générer le code qui calcule cette carte de contact, puis de généraliser cette carte entre une protéine et un ligand.

But du projet :

On se propose dans un premier temps de générer une carte de contact de la protéine, d'abord sous forme binaire (absence/présence de contact), puis avec un code couleur dépendant de la distance. Si ces premières étapes sont fructueuses, on pourra étendre la carte à la protéine accompagnée d'un ligand (si ce dernier a été cristallisé avec la protéine), constituant un moyen simple de visualiser où le ligand fait des contacts avec la protéine. La matrice pourra être générée et stockée avec la bibliothèque `numpy` puis être affichée sous forme d'image avec la bibliothèque `matplotlib`. (sous Windows, `numpy` et `matplotlib` s'installent en même temps que python lui-même si vous téléchargez la dernière version 2.7.11 accessible sur https://www.python.org/downloads/).

Etapes suggérées :

1. D'abord tenter la carte de contact binaire (présence / absence de contact) sur la protéine seule en utilisant uniquement les atomes C-alpha :
    - Il s'agit de calculer les distances entre toutes les paires de C-alpha de la protéine ; on démarrera avec une protéine monomérique, par exemple 1u59 (Tyrosine Kinase)
    - Pour simplifier, on considère que l'on a un contact si la distance C-alpha / C-alpha est inférieure à 8 Å. (constante à mettre en haut du script python, par exemple : `SEUIL = 8`) (vous pouvez vous amusez à tester différents seuils de contact)
    - Dans un premier temps, je vous conseille de générer une liste à l'écran indiquant les distances C-alpha / C-alpha, par exemple :

```
[...]
LYS  328 - ARG  400 :  16.06 angstroms --> pas de contact
[...]
SER  436 - ALA  601 :   5.42 angstroms --> contact !
[...]
```

      Bien vérifier que vos distances sont correctes avec `pymol` ou un autre visualiseur de structure.
    - Générez ensuite une matrice avec numpy et la fonction `array` (0 = absence de contact, 1 = présence de contact), vous pouvez l'écrire dans un fichier avec la fonction `numpy.savetxt()`. Puis créez une image de cette matrice avec matplotlib et la fonction `matplotlib.matshow()` (voir par exemple la page http://matplotlib.org/examples/pylab_examples/matshow.html)

CONSEIL : commencez sur un exemple simple, par exemple les 10 ou 20 premiers résidus de la protéine, et bien vérifier vos résultats au fur et à mesure.

Pour aller plus loin… :

2. Tentez la même carte de contact mais en mettant un niveau de gris entre 0 et 1 inversement proportionnel à la distance (au-delà de 15 Å on peut mettre à 0). N'oubliez pas de préciser le code couleur utilisé (en mettant une échelle sur le côté idéalement, vous pouvez vous inspirer de ce code : http://stackoverflow.com/questions/21071128/matplotlib-plot-numpy-matrix-as-0-index).
3. Réalisez une carte de contact comme en (1) mais en incluant les atomes C-alpha de la protéine **et** l'ensemble des atomes du ligand (d'abord la protéine, puis le ligand).

Consignes :

- Votre code devra respecter les conventions PEP8 et celles que nous avons vues en cours.
- Le code sera lancé de la manière suivante en passant le fichier PDB comme premier argument, le type d'analyse et de plot souhaité comme 2ème argument (`ProtBin`, `ProtColor` ou `ProtLig` pour les méthodes 1, 2 ou 3 décrites ci-dessus), puis le nom du ligand dans le fichier PDB comme 3ème argument, par exemple :
  ```
  python ./contact_map.py 1u59.pdb ProtBin STU
  ```
- Les contacts seront écrits sous forme de liste (cf ci-dessus) dans un fichier `1u59_liste_contacts.txt`, la matrice de contacts au format texte sera écrite dans un fichier `1u59_matrice_contacts.dat` et la carte de contact sera sauvée dans le fichier image `1u59_carte_contacts.png`. (si vous utilisez un autre fichier PDB, `1u59` sera remplacé par le code PDB du nouveau fichier)

Conseils :

- Pensez à bien vérifier vos résultats. Un programme renvoie toujours un résultat, nous voulons que votre programme renvoie le bon résultat.
- **Comme dit ci-dessus, commencez avec un fichier PDB simplifié qui contient seulement 10 à 20 résidus.** Vérifiez sur cet exemple simple que tout fonctionne comme attendu.
- Pour la version avec ligand, pensez au site https://www.ebi.ac.uk/pdbsum qui fait le bilan des contacts ligand / protéine pour chaque fichier dans la PDB.
- Pour parser un fichier PDB : http://cupnet.net/pdb-format/.
- Ne vous lancez pas trop rapidement dans le code, faites un plan du découpage en fonctions / programme principal. Notre conseil, vous pouvez organiser votre code en fonctions en vous posant (dans l'ordre) les questions suivantes :
  - Quelles sont les données à récupérer à partir du fichier d'entrée ? Les arguments passés par l'utilisateur sont-ils valides ?
  - Quelles sont les opérations / calculs / analyses à faire à partir de ces données ?
  - Comment afficher le résultat de ces opérations effectuées par mon programme ? Vers quelle sortie (fichier / écran / image) dois-je renvoyer les résultats de mon programme ?
- Pensez à l'écriture formatée pour la sortie.
- Vous pouvez vous entraider pour les raisonnements, mais il s'agit d'un travail personnel avant tout. Nous nous rendrons compte tout de suite des copier/coller de code.
- Ne copiez pas de code depuis internet sans le comprendre et le maitriser parfaitement.
- Si votre PDB ne commence pas au résidu 1, attention aux axes de la matrice.