Synthèse

Introduction

La grippe est une maladie infectieuse et contagieuse provoquée par le virus influenza.
Le mode d'action de ce virus est similaire aux autres virus, c'est-à-dire qu'il va pénétrer la cellule pour y déverser son matériel génétique et ainsi obtenir, via la machinerie de réplication de la cellule hôte, les protéines virales nécessaire au bon fonctionnement du virus.

Pour lutter contre ce virus, on peut choisir d'inhiber une de ces protéines virales afin de perturber le développement du virus et ainsi diminuer sa prolifération.

On s'intéresse ici à la protéine NS1 du virus influenza.

NS1

La protéine NS1 du virus influenza a deux domaines, un domaine de liaison à l'ARN (le RBD), et un domaine effecteur. Ces deux domaines sont liés par un linker. Le RBD est la zone clé de cette protéine car pour remplir sa fonction, le RBD doit être lié à de l'ARN.

Objectifs

Notre but est donc de bloquer la liaison de l'ARN avec le RBD pour ainsi empêcher la protéine NS1 de remplir ses fonctions qui favorisent la prolifération du virus influenza et donc favorisent l'apparition de la maladie.

Pour ce faire, nous avons choisi d'utiliser un ligand qui va se fixer sur le RBD et ainsi, l'ARN ne pourra pas se fixer sur le RBD et on aura inhiber la NS1.

Nous allons avoir besoin de plusieurs outils pour mener à bien ce travail.

Matériels & Méthodes

Voici la liste des outils que nous allons utiliser pour traiter ce problème : Python, Pymol, PockDrug, AutoDock Tools, Vina.

Outils

Python

Python est un outil indispensable car il va nous permettre d'automatiser la grande majorité des tâches qui, manuellement, prennent beaucoup de temps et sont surtout très répétitives.

On peut s'en servir pour effectuer un traitement sur une conformation de la protéine NS1 et l'appliquer pour toutes les autres conformations tout en gérant les données obtenues.

Pymol

Pymol va nous servir à visualiser la protéine ou les différentes conformations de la protéine NS1 pour l'analyser.

On peut l'utiliser dans des scripts Python, ce qui est très utile quand on veut automatiser une tâche dans Pymol (comme détecter la position du ligand après un docking).

PockDrug

Chaque protéine a un certain nombre de poches qui sont des cavités où un ligand peut se loger. PockDrug peut détecter ces poches et nous en donner des caractéristiques comme la Druggabilité qui correspond à une valeur de « réussite » pour une poche, plus elle est élevée, plus l'affinité entre le ligand et cette poche est élevée.

ADT

Pour effectuer un docking entre un récepteur et un ligand, nous devons les préparer. Pour cela, on utilise AutoDock Tools pour ajouter des caractéristiques indispensables au docking sur les protéines/ligands pour que tout se passe correctement.

A l'aide d'ADT, on a aussi la possibilité de cibler la zone de docking sur une partie de la protéine. Pour notre cas, nous avons choisi d'effectuer un docking en aveugle et de regarder où se fixe le ligand, pour cela on définit une zone de docking (Grid Box) qui enveloppe toute la protéine, ainsi, le ligand pourra se fixer sur toute la protéine selon les poches disponibles.

Vina

Avec Vina, on effectue le docking à proprement parler. Ce logiciel prend plusieurs paramètres comme la position de la Grid Box (soit la zone de docking), le récepteur, le ligand…

Protocole

PockDrug

Nous devons, dans un premier temps, récupérer la liste des poches disponibles pour les 68 conformations différentes de NS1 obtenues grâce à la dynamique moléculaire qui simule les variations de la structure d'une protéine dans le temps.

Pour cela, on doit charger les 68 conformations sur le site PockDrug et télécharger le dossier crée par ce dernier qui contient les poches possibles pour chaque conformation.

On s'aide d'un script Python qui va charger une à une les conformations dans PockDrug, attendre que le site fasse son analyse, puis le script va télécharger le dossier, pour chaque conformation, qui contient les poches.

Docking

Préparation

Pour réaliser un docking d'un ligand sur une protéine, nous devons préparer les deux acteurs.
On réalise la préparation avec le logiciel AutoDock Tools qui va ajouter certaines caractéristiques sur la protéine ou ligand.

Pour la protéine, on doit : Enlever l'eau, ajouter les hydrogènes polaires et les charges de Gasteiger.

On choisit la zone de docking avec l'option de la Grid Box et on sauvegarde la position de la Box pour pouvoir l'utiliser avec Vina.

Pour le ligand, il faut d'abord passer sur MarvinSketch afin de rendre le ligand, initialement 2D, en 3D. Ensuite, via ADT, on ajoute les charges de Gasteiger et les axes de rotations de la molécule.

Ces caractéristiques sont nécessaires pour que le docking se passe correctement.

Vina

Nous avons précédemment positionné la Grid Box autour de la protéine et on a sauvegarder les coordonnées de la Box.

On va maintenant écrire ces coordonnées dans un fichier texte ainsi que d'autres paramètres comme le nom du récepteur, du ligand, du fichier de sortie crée suite au docking, du fichier log qui va contenir toutes les informations relatives au docking…

Ce fichier texte est un fichier de configuration utilisé par Vina pour réaliser le docking.

Une fois le docking fini, nous avons à disposition un nouveau fichier en format .pdbqt qui contient le ligand docké sur la protéine.

Pymol

On peut visualiser le docking à l'aide de Pymol. On ouvre la protéine (par exemple la conformation *0002*), et on ouvre le fichier crée par Vina qui contient le ligand docké (par exemple *output_0002.pdbqt*).

On peut ainsi vérifier dans quelle poche le ligand s'est fixé. Il faut le vérifier pour les 68 conformations, pour cela, on s'aide de scripts Python.

Python

Un premier script Python va calculer la distance entre le centre d'inertie du ligand et celui de toutes les poches d'une conformation et retenir la distance la plus faible comme étant la poche qui contient le ligand.

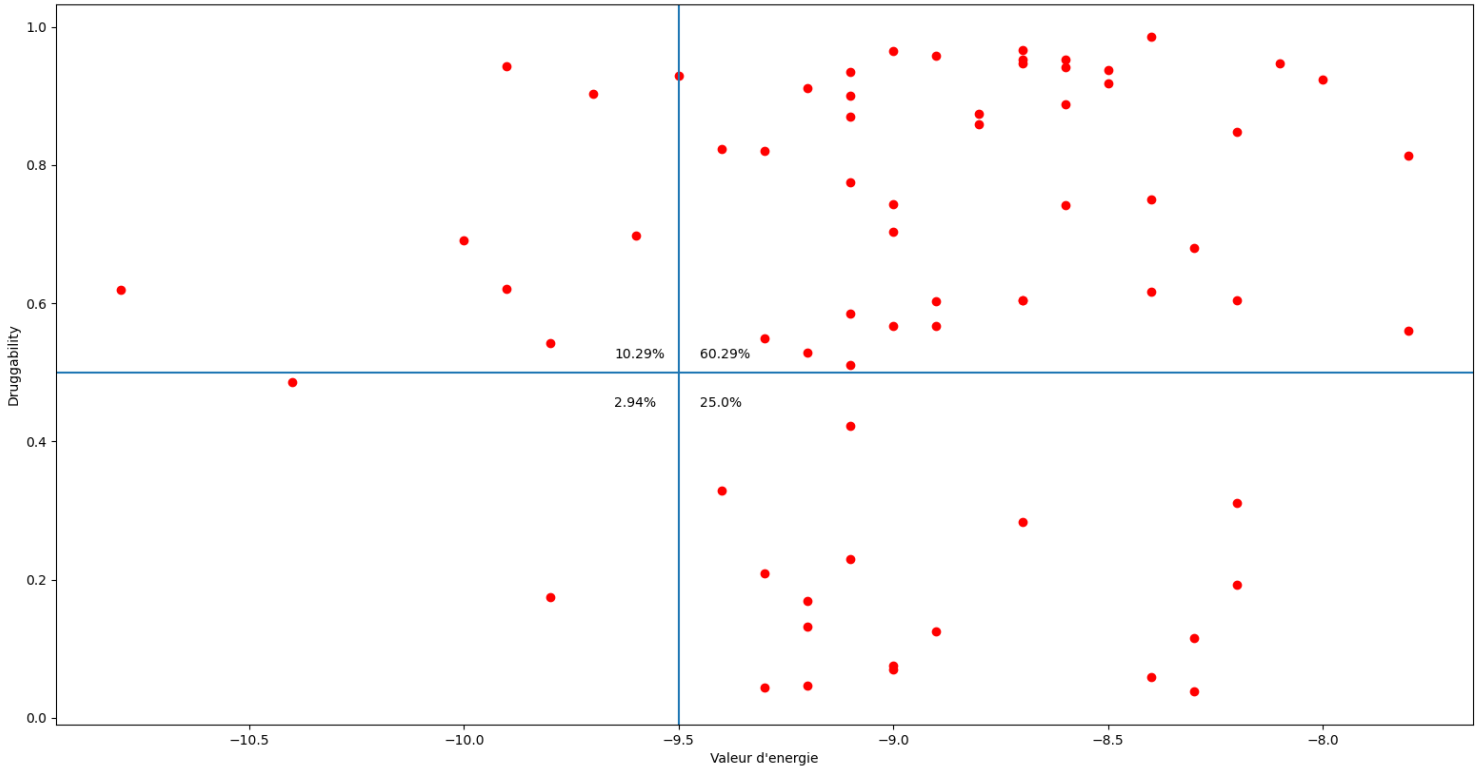
On répète ce processus pour toutes les conformations.

Un deuxième script Python va retenir la valeur d'énergie de la poche qui contient le ligand (valeur donnée par Vina suite au docking) et la druggabilité de cette poche (valeur donnée par PockDrug).

Nous avons donc pour une conformation : le nom de la poche qui contient le ligand, l'énergie et la druggabilité de cette poche.

On fait de même pour toutes les conformations, et à la fin, nous avons des données que l'on peut représenter sous forme de graphique.

Résultat



Sur ce premier graphique est représenté en ordonné la druggabilité et en abscisse la valeur d'énergie d'une poche. Les points en rouge représentent les poches qui contiennent le ligand.

La zone qui nous intéresse est la partie en haut à gauche car la druggabilité est assez élevée (>0.5) et la valeur d'énergie est basse ce qui est synonyme d'une bonne stabilité entre le ligand et la poche.

Ce graphique nous donne une vision globale des données obtenues avec les dockings.

On va maintenant se focaliser sur le RBD qui est le centre de notre problème. Au lieu de retenir toutes les conformations pour un graphique, nous allons prendre uniquement celles qui ont le ligand fixé dans le RBD.

Pour cela, on demande au script Python de retenir les conformations qui ont une distance entre ligand et poche inférieure à 20.

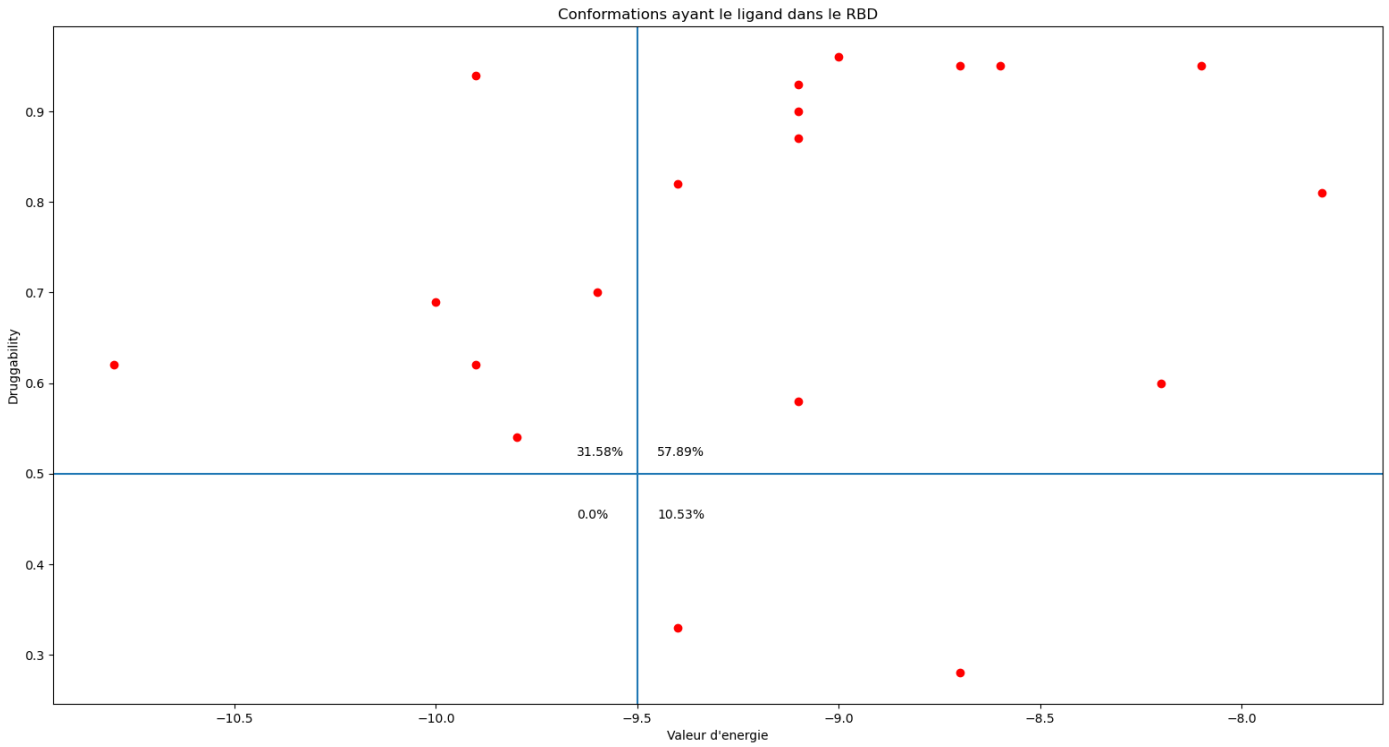
On a comme sortie de script :

```
Conformation     Distance                        Energie

0108             8.028900187123854               -9.6
0029             8.03720211160448                -10.0
0138             8.228623311981774               -9.1
0120             8.913586385710822               -9.9
0116             8.946120768032088               -9.4
0040             9.539624351422187               -10.8
0101             9.839053341386665               -8.1
0140             9.923828334906652               -9.1
0036             10.074803521077287              -9.1
0134             10.765545342080342              -8.6
0015             10.881350078496505              -9.4
0045             10.934097425098905              -8.7
0112             11.498780407549459              -8.7
0139             11.818069011772215              -9.1
0026             11.945243938972196              -9.8
0035             11.949122712863211              -7.8
0030             13.007174140287411              -9.9
0019             14.793329457994222              -9.0
0018             15.120948954746998              -8.2

Nombre de conformations ayant le ligand proche du résidu 38 :  19
```

Ces 19 conformations ont le ligand proche du résidu 38. On réalise alors un graphique avec ces conformations :



Ici également, la partie qui nous intéresse est la partie en haut à gauche car on a une bonne druggabilité et une bonne valeur d'énergie.

Les conformations qui se trouvent en haut à gauche sont :

```
PARTIE EN HAUT A GAUCHE

Energie          Druggability          Conformation    Poche
-10.8            0.62                  0040            pocket0
-10.0            0.69                  0029            pocket0
-9.9             0.62                  0030            pocket0
-9.9             0.94                  0120            pocket0
-9.8             0.54                  0026            pocket0
-9.6             0.7                   0108            pocket0

Nombre de conformations :  6
```

Donc pour traiter notre problème de départ, il faudrait se focaliser sur ces conformations car le ligand se trouve dans le RBD suite à un docking en aveugle.

Conclusion

On a une liste de conformations qui ont le ligand dans le RBD suite à un docking en aveugle, mais la méthode utilisée a quelques défauts notamment sur la fiabilité du script qui détecte l'endroit du docking avec les calculs de distances entre le centre d'inertie du ligand et de toutes les poches.

En effet, il y a des cas où la distance entre ligand et poche est la plus faible mais cela ne veut pas tout le temps dire que le ligand se trouve dans cette poche. Elle peut se trouver dans la poche A mais être plus proche de la poche B que de la A, donc le script nous dira que le ligand se trouve dans la poche B.

Le script n'a donc pas une précision de 100%.

Un critère à ajouter serait de vérifier les séquences, c'est-à-dire vérifier si les résidus qui se trouvent autour du ligand sont aussi dans la poche détecté par le script, mais sur Pymol, la fonction align donne des résultats peu compréhensibles, donc on ne peut pas vraiment faire confiance à cette fonction qui peut donner un gros score d'alignement pour une poche A alors que celle-ci ne contient pas de résidus proche du ligand, et donner un score plus faible à une poche B alors que celle-ci contient des résidus qui sont proches du ligand et donc on devrait obtenir le meilleur alignement.

Cela est peut-être dû à un manque de compréhension de ce fait exactement la fonction align.

Bibliographie

The Influenza A Virus Protein NS1 Displays Structural Polymorphism
Berenice Carrillo, Jae-Mun Choi, Zachary A. Bornholdt, Banumathi Sankaran, Andrew P. Rice, B. V. Venkataram Prasad
Journal of Virology Mar 2014, 88 (8) 4113-4122; DOI: 10.1128/JVI.03692-13

Engel, D. A. (2013). **The influenza virus NS1 protein as a therapeutic target**. *Antiviral research*, *99*(3), 409-416.

The influenza virus NS1 protein as a therapeutic target
Daniel A.Engel
https://doi.org/10.1016/j.antiviral.2013.06.005

AutoDock Vina Video Tutorial
www.youtube.com/watch?v=-GVZP0X0Tg8

Tutorial : site specific docking using auto dock vina.
www.youtube.com/watch?v=blxSn3Lhdec&list=WL&index=33&t=0s

AutoDock Vina Tutorial
https://www.youtube.com/watch?v=rBEKZQ22nhs&list=WL&index=32& t=3257s

Selenium (Module Python) Documentation
https://selenium-python.readthedocs.io/index.html

Annexes

Liste des scripts (à copier-coller dans un éditeur de texte)

Les scripts pythons sont à mettre dans un même dossier car plusieurs scripts travaillent ensembles.

Réaliser tous les dockings avec Vina

Ce que fait le script :

- C'est un script bash qui va lancer les dockings avec Vina pour toutes les conformations et créer un dossier pour chaque docking/conformations.

```
for f in conf/*;
do

    b=`basename $f .pdbqt`
    echo Processing ligand $b
    mkdir -p ./result/$b


    ./vina --config dock.txt --receptor $f --out ./result/$b/out_$b.pdbqt --log ./result/$b/log_$b.txt

done;
```

Télécharger les poches sur PockDrug pour toutes les conformations

Ce que fait le script :

- Va sur PockDrug
- Charge une conformation
- Ouvre d'autres onglets pour charger d'autres conformations
- Après avoir charger un certain nombre X (qu'on peut choisir afin d'éviter une surcharge d'onglets), le script va retourner sur le premier onglet, et si l'analyse par PockDrug est terminée, il procède au téléchargement du dossier contenant les poches pour la conformation en cours (l'onglet actuel) et ferme l'onglet puis passe à l'onglet suivant pour télécharger les autres dossiers.
  On peut spécifier le dossier cible pour le téléchargement.
- Une fois qu'on a téléchargé X dossiers, le script charge les X prochaines conformations et procède de la même manière.
  Afin de ne pas avoir à charger les 68 conformations, on peut en charger par tranche de 10 et afin d'éviter la surcharge, les onglets sont progressivement fermés.

Comment l'utiliser :

- Les seuls paramètres à modifier par l'utilisateur sont dans l'encadré délimité par des « **####** »
  Principalement des chemins d'accès à des **dossiers**, mais aussi le nombre total **d'onglets** qu'on souhaite ouvrir, et la valeur de la **tranche**
  Il faut également spécifier le chemin d'accès au **driver** (qui dépend du navigateur utilisé) donnée par **Selenium** (le module utilisé pour automatiser les tâches des navigateurs)

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import os



#CHEMINS
############################################################


#DOSSIER CIBLE POUR LES TELECHARGEMENTS
download_dir = '/home/ragou/Bureau/NS1/conf/pockdrug'


#DOSSIER CONTENANT LES CONFORMATIONS A TESTER
conformation = '/home/ragou/Bureau/NS1/conf/PDB_NS1'


#CHEMIN QUI MENE AU WEBDRIVER DE VOTRE NAVIGATEUR
driver = "/home/ragou/Bureau/NS1/chromedriver"


############################################################


#PARAMETRES
############################################################
```

```
#NOMBRE DE FICHIER QUI VONT S'OUVRIR AU TOTAL
onglet_total = 68

#TELECHARGEMENT PAR TRANCHE DE
tranche = 10


#################################################################




onglet_ouvert = 0
indice_depart = 0
nb_fichier_ouvert = 0

dossier = os.listdir(conformation)

filename = []


downname = "pocket_PPE.zip"


#OPTIONS POUR DEFINIR LE DOSSIER CIBLE DES TELECHARGEMENT
chrome_options = webdriver.ChromeOptions()
chrome_options.add_experimental_option("prefs",
{"download.default_directory": download_dir })


#OUVRE UNE PAGE CHROME
navigateur = webdriver.Chrome(chrome_options = chrome_options,
executable_path = driver)
navigateur.maximize_window()

#ALLER SUR LE SITE
navigateur.get("http://pockdrug.rpbs.univ-paris-diderot.fr/cgi-
bin/index.py?page=Druggability")

#POUR METTRE LES CONFORMATIONS SUR POCKDRUG
for fichier in dossier:


      nb_fichier_ouvert +=1
      onglet_ouvert += 1


      #SI ON A ATTEINT LE NOMBRE DE FICHIER OUVERT VOULU (onglet) ON STOP
      if nb_fichier_ouvert > onglet_total:
            navigateur.close()
            nb_fichier_ouvert -= 1
            break


      print("Conformation ouverte : {}".format(fichier))

      filename.append(fichier)
```

```
        upload = '{}/{}'.format(conformation,fichier)

        #CHOISIR LA 2E SECTION
        navigateur.find_element_by_xpath('/html/body/div[2]/div/div[1]/div[2]
/form/fieldset[2]/div/label/font/b/input').click()


        #COCHER "UPLOAD YOUR PDB FILE"
        navigateur.find_element_by_id("Radio1").click()

        #UPLOAD LE FICHIER
        navigateur.find_element_by_xpath('/html/body/div[2]/div/div[1]/div[2]
/form/div[2]/div/fieldset[1]/div/div[1]/a/input').send_keys(upload)

        #SUBMIT
        navigateur.find_element_by_xpath("/html/body/div[2]/div/div[2]/center
/input[1]").click()

        #OUVRIR UN NOUVEL ONGLET
        navigateur.execute_script("window.open('http://pockdrug.rpbs.univ-
paris-diderot.fr/cgi-bin/index.py?page=Druggability')")


        if onglet_ouvert == tranche or nb_fichier_ouvert == onglet_total:
                print("\n\n")

                #TELECHARGE LES POCHES DES CONFORMATIONS OUVERTS DANS POCKDRUG
                for i in range(indice_depart,indice_depart + onglet_ouvert):


                        navigateur.switch_to.window(navigateur.window_handles[0])
                        time.sleep(1)

                        navigateur.find_element_by_xpath('//*[@id="PPE-
desc"]/div[2]/form/div/input').click()
                        navigateur.close()

                        time.sleep(2)

        os.rename("{}/{}".format(download_dir,downname),"{}/{}.zip".format(do
wnload_dir,os.path.splitext(filename[i])[0]))

                print("Avancee : {}\n".format(nb_fichier_ouvert))
                indice_depart += onglet_ouvert
                onglet_ouvert = 0



        #ON SWITCH SUR LE NOUVEL ONGLET POUR POUVOIR AGIR DESSUS
        navigateur.switch_to.window(navigateur.window_handles[onglet_ouvert])



print("\nNombre de fichiers telecharges au total :
{}".format(nb_fichier_ouvert))
```

Détecter les poches qui ont le ligand

Ce que fait le script :

- Charge la conformation mis en argument de la fonction detect_distance
- Charge le résultat de docking lié à cette conformation, soit le ligand docké
- Récupère la position du ligand
- Récupère la position d'une première poche
- Calcul la distance entre le ligand et la poche
- Si elle est inférieure à un certain seuil, on retient cette poche et cette distance devient le nouveau seuil afin de réduire progressivement ce seuil pour trouver la poche avec la distance la plus faible
- On charge les autres poches et on fait de même

Comment l'utiliser :

- Il suffit de spécifier le chemin vers le dossier contenant les résultats de docking (**path_dock**) et le dossier contenant les poches téléchargées via PockDrug (**path_poche**)

```
from pymol import cmd
from math import sqrt
import os

#FONCTION POUR DETECTER LA POCHE QUI CONTIENT LE LIGAND

#CONFORMATION : CONFORMATION POUR LAQUELLE IL FAUT DETECTER LA POCHE QUI
CONTIENT LE LIGAND
#PATH_DOCK : CHEMIN QUI MENE AU DOSSIER QUI CONTIENT LES RESULTATS DE
DOCKINGS
#PATH_POCHE : CHEMAIN QUI MENE AU DOSSIER QUI CONTIENT LES POCHES POUR LA
CONFORMATION

def detect_distance(conformation ,path_dock,path_poche):

    #DOSSIER QUI CONTIENT LES POCHES
      dossier_poche = os.listdir(path_poche)

    #DOSSIER QUI CONTIENT LES RESULTATS DE DOCKING
      dossier_dock =  os.listdir(path_dock)



    #DOSSIER QUI CONTIENT LES CONFORMATIONS
      dossier_conf = os.listdir("{}/{}".format(path_poche,conformation))


      #ON CHARGE DANS PYMOL LA CONFORMATION ET LE LIGAND
      cmd.load("{}/{}/{}.pdbqt".format(path_dock,conformation,conformation)
,"conf")
      cmd.load("{}/{}/out_{}.pdbqt".format(path_dock,conformation,conformat
ion),"lig")



    #ON SE FOCALISER SUR LA PREMIERE POSE DU LIGAND
      cmd.create("obj","lig",1)
      cmd.center("obj")
```

```
    #ON SAUVEGARDE LA POSITION DU CENTRE D'INERTIE DU LIGAND
      poz_lig = cmd.get_position()


    #VALEUR DE REFERENCE POUR LA DISTANCE QUI CHANGE AU FUR ET A MESURE
      D_ref = 100
      pock = ''

    #POUR TOUTES LES POCHES DE LA CONFORMATION
      for i in range(len(dossier_conf) - 1):

        #ON CHARGE LA POCHE DANS PYMOL
            nom = "pocket{}".format(i)

      cmd.load('{}/{}/{}_atm.pdb'.format(path_poche,conformation,nom),nom)
            cmd.center(nom)

        #ON SAUVEGARDE LA POSITION DU CENTRE D'INERTIE DE LA POCHE
            poz_pock = cmd.get_position()

        #ON CALCULE LA DISTANCE ENTRE LE CENTRE D'INERTIE DE LA POCHE ET DU
LIGAND
            D = sqrt((poz_lig[0]-poz_pock[0])**2 +(poz_lig[1]-
poz_pock[1])**2 +(poz_lig[2]-poz_pock[2])**2)

        #ON RETIENT LES POCHES QUI ONT UNE DISTANCE PAR RAPPORT AU LIGAND
INFERIEUR A UNE DISTANCE REFERENCE
            if(D < D_ref):

            #ON ACTUALISE LA DISTANCE DE REFERENCE A LA DISTANCE LA PLUS
FAIBLE
                  D_ref = D
                  pock = nom



            cmd.delete(nom)

      cmd.delete("conf")
      cmd.delete("lig")
      print("La meilleure poche : {} pour le docking
{}".format(pock,conformation))

      return {conformation:pock}




path_dock = "C:/Users/Radja/Desktop/NS1/Vina/result"
path_poche = "C:/Users/Radja/Desktop/NS1/conformation/pockdrug"
dossier_dock =  os.listdir(path_dock)
i = 0

for conformation in dossier_dock:


      detect_distance(conformation,path_dock,path_poche)
```

Détecter les conformations qui ont le ligand proche du résidu 38

Ce que fait le script :

- Charge une conformation
- Charge le résultat de docking associé à cette conformation, soit le ligand
- Récupère la position du ligand
- Récupère la position du résidu 38 qui se trouve dans le RBD
- On retient toutes les conformations qui ont une distance entre le ligand et le résidu 38 inférieure à un certain seuil, nous avons choisi 20 pour ce script car toutes les conformations qui ont une distance ligand-residu38 en dessous de 20 ont le ligand dans le RBD.

Comment l'utiliser :

- Il suffit de spécifier le chemin vers le dossier contenant les résultats de docking (**path_dock**)

```
from pymol import cmd
from math import sqrt
import os

#FONCTION POUR DETECTER LES CONFORMATIONS QUI ONT LE LIGAND PROCHE DU
RESIDU 38 (= DANS LE RBD)
def detect_res38(path_dock,distance,pic = 0):

      dossier_dock =  os.listdir(path_dock)
      ok = {}


      for conformation in dossier_dock:


      cmd.load("{}/{}/{}.pdbqt".format(path_dock,conformation,conformation)
,"conf")

      cmd.load("{}/{}/out_{}.pdbqt".format(path_dock,conformation,conformat
ion),"lig")



            cmd.create("obj","lig",1)

            cmd.center("obj")
            poz_lig = cmd.get_position()


            cmd.select("r38","res 38")
            cmd.center("r38")

            poz_38 = cmd.get_position()


            D = sqrt((poz_lig[0]-poz_38[0])**2 +(poz_lig[1]-poz_38[1])**2
+(poz_lig[2]-poz_38[2])**2)
```

```
            if(D < distance):

                  if(pic == 1):
                        cmd.show("surface","r38")
                        cmd.set_view ("\
                            -0.298310757,    0.680330575,   -0.669443905,\
                             0.016840594,    0.705018342,    0.708981633,\
                             0.954314768,    0.200224459,   -0.221768484,\
                            -0.000218070,   -0.000902027, -86.197776794,\
                            40.384254456,   78.192932129,   43.972404480,\
                            -5.481759548,  177.321334839, -20.000000000" )
                        cmd.png(conformation,width = 1700, height = 750)
                  ok.update({conformation:D})

            cmd.delete("conf")
            cmd.delete("lig")


      ok = sorted(ok.items(),key= lambda t:t[1])

      return ok




path_dock = "C:/Users/Radja/Desktop/NS1/Vina/result"


dossier_dock =  os.listdir(path_dock)


ok = detect_res38(path_dock,20)

print("\nConformation \t Distance \t\t\t Energie\n")
for conf, dis in ok:
      nrj = energie.valeur_energie(path_dock,conf,"1")
      print(conf,"\t\t",dis,"\t\t",nrj)

print("\nNombre de conformations ayant le ligand proche du résidu 38 :
",len(ok))
```

Récupérer les valeurs d'énergie et de Druggabilité pour une poche et réalisation des graphiques

Ce que fait le script :

- La fonction valeur_energie va récupérer la valeur d'énergie d'une poche (celle qui correspond à la pose 1) dans le fichier log qui provient de Vina suite au docking. Dans ce fichier texte, nous avons plusieurs poses qui correspondent à différentes positions du ligand après docking classées selon la valeur d'énergie (la pose 1 étant la meilleure, la valeur d'énergie est la plus basse)
- La fonction valeur_drug va récupérer la valeur de druggabilité d'une poche dans le fichier texte fourni par PockDrug qui résume les différentes caractéristiques des poches avec notamment la valeur de druggabilité de celle-ci.
- Pour toutes les conformations, on récupérer le nom de la poche qui contient le ligand à l'aide de la fonction qui calcule les distances ligand-poche, puis on lance les deux fonctions pour récupérer l'énergie et la druggabilité de cette poche
- Puis, quelques lignes de codes pour la mise en forme des données obtenues (calcul de pourcentages, affichage…)
- Enfin, le graphique à l'aide du module **matplotlib**

Comment l'utiliser :

- Il suffit de spécifier le chemin vers le dossier contenant les résultats de docking (**path_dock**) et le dossier contenant les poches téléchargées via PockDrug (**path_poche**)

```
import os
import detect
import matplotlib.pyplot as plt
import res38

#FONCTION POUR RECUPERER LA VALEUR D'ENERGIE D'UNE POCHE
def valeur_energie(path_dock,conformation,pose):

      with
open("{}/{}/log_{}.txt".format(path_dock,conformation,conformation)) as
fich:
            for ligne in fich:
                  if(ligne[:4] =='   {}'.format(pose)):
                        mot = ligne.split(" ")

                        if(mot[11] ! = ""):
                              return (float(mot[11]))
                        else:
                              return (float(mot[12]))



#FONCTION POUR RECUPERER LA VALEUR DE DRUGGABILITE D'UNE POCHE
def valeur_drug(path_poche,conformation,poche):
      nom = "{}_atm".format(poche)

      with open("{}/{}/pocket_PPE.txt".format(path_poche,conformation)) as
fich:
            for ligne in fich:
                  mot = ligne.split("\t")
```

```
                if(mot[0] == nom):

                    return (float(mot[-2]))


path_dock = "C:/Users/Radja/Desktop/NS1/Vina/result"
path_poche = "C:/Users/Radja/Desktop/NS1/conformation/pockdrug"

dossier_poche = os.listdir(path_poche)
dossier_dock =  os.listdir(path_dock)




dico = {}
pose = "1"

energie = []
drug = []
conf_list = []
poche_list= []




#POUR LES CONFORMATIONS AYANT LE LIGAND PROCHE DU RESIDU 38

"""
proche_res38 = res38.detect_res38(path_dock,20)

for conf, dis in proche_res38:
      dico.update(detect.detect_distance(conf,path_dock,path_poche))
"""

#POUR TOUTES LES CONFORMATIONS
for conf in dossier_dock:
      dico.update(detect.detect_distance(conf,path_dock,path_poche))



#ON SAUVEGARDE LES VALEURS D'ENERGIE ET DE DRUGGABILITE
for conformation,poche in dico.items():
      conf_list.append(conformation)
      poche_list.append(poche)
      energie.append(valeur_energie(path_dock,conformation,pose))
      drug.append(round(valeur_drug(path_poche,conformation,poche),2))

energie,drug,conf_list,poche_list =
zip(*sorted(zip(energie,drug,conf_list,poche_list)))

taille = len(energie)



plt.plot(energie,drug,'ro')
plt.ylabel("Druggability")
plt.xlabel("Valeur d'energie")
plt.axvline(-9.5)
```

```
plt.axhline(0.5)

list_HG = []
list_BG = []
list_HD = []
list_BD = []
HG = 0
HD = 0
BG = 0
BD = 0

#COMPTAGE DES POINTS ROUGES DU GRAPHIQUE SELON LA ZONE (HG : HAUT_GAUCHE …)
for e,d,con,po in zip(energie,drug,conf_list,poche_list):
      if(e < -9.5 and d > 0.5):

            list_HG.append((e,d,con,po))
            HG +=1

      if(e < -9.5 and d < 0.5):
            list_BG.append((e,d,con,po))
            BG +=1

      if(e > -9.5 and d > 0.5):
            list_HD.append((e,d,con,po))
            HD +=1

      if(e > -9.5 and d < 0.5):
            list_BD.append((e,d,con,po))
            BD +=1


#CALCUL DE POURCENTAGE POUR CHAQUE ZONE

PHG = round(HG/taille * 100,2)
PHD = round(HD/taille * 100,2)
PBG = round(BG/taille * 100,2)
PBD = round(BD/taille * 100,2)


#MISE EN FORME DES DONNEES SELON LA ZONE

print("\nPARTIE EN HAUT A GAUCHE\n")
print("Energie\t\t Druggability\t\t Conformation\t Poche")

for e,d,c,p in list_HG:

      print(e,'\t\t',d,'\t\t\t',c,'\t\t',p)
print("\nNombre de conformations : ",len(list_HG))


print("\nPARTIE EN HAUT A DROITE\n")
print("Energie\t\t Druggability\t\t Conformation\t Poche")

for e,d,c,p in list_HD:

      print(e,'\t\t',d,'\t\t\t',c,'\t\t',p)
print("\nNombre de conformations : ",len(list_HD))



print("\nPARTIE EN BAS A GAUCHE\n")
```

```
print("Energie\t\t Druggability\t\t Conformation\t Poche")

for e,d,c,p in list_BG:

      print(e,'\t\t',d,'\t\t\t',c,'\t\t',p)
print("\nNombre de conformations : ",len(list_BG))


print("\nPARTIE EN BAS A DROITE\n")
print("Energie\t\t Druggability\t\t Conformation\t Poche")

for e,d,c,p in list_BD:

      print(e,'\t\t',d,'\t\t\t',c,'\t\t',p)
print("\nNombre de conformations : ",len(list_BD))




#GRAPHIQUE

plt.annotate("{}%".format(PHG),xy=(-9.65,0.52))

plt.annotate("{}%".format(PHD),xy=(-9.45,0.52))

plt.annotate("{}%".format(PBG),xy=(-9.65,0.45))

plt.annotate("{}%".format(PBD),xy=(-9.45,0.45))

#plt.title("Conformations ayant le ligand dans le RBD")
plt.show()
```