Comparaison de la variabilité structure des ligands chez PR1 et PR2

Leslie REGAD et Maxime KERMARREC

2019-02-27

Contents

Commencement du projet : vendredi 01/03/2019

Objectifs

Etudier la flexibilité des atomes des ligands chez PR1 et PR2

Données

- fichier pdb des ligands extraits des structures de PR1 et PR2 superposée.
  Ces fichiers se trouve dans le réperotoire `data/ligands_pdb_cleaned_file_ATOM`
- type de PR :

PDB code	type	remarque
3s45	PR2	
1hhp	PR1	(monomère)
1hih	PR1	
1hii	PR2	
1hiv	PR1	
1hpv	PR1	
1hsh	PR2	
1hsi	PR2	
1ivp	PR2	
1sdt	PR1	(peu avoir des pbl de numérotation des résidus)
2hb3	PR1	
2hb4	PR1	(monomère)
2hpe	PR2	
2hpf	PR2	
2ien	PR1	(peu avoir des pbl de numérotation des résidus)

PDB code	type	remarque
2mip	PR2	
2nph	PR1	(peu avoir des pbl de numérotation des résidus)
3phv	PR1	(monomère)
2z4o	PR1	(peu avoir des pbl de numérotation des résidus)
3ebz	PR2	(peu avoir des pbl de numérotation des résidus)
3ec0	PR2	(peu avoir des pbl de numérotation des résidus)
3ecg	PR2	(peu avoir des pbl de numérotation des résidus)
3ekv	PR1	
3nu3	PR1	(peu avoir des pbl de numérotation des résidus)
4hla	PR1	
4ll3	PR1	
1hsh	PR2	

```
Proteases = c("1hhp","1hih","1hii","1hiv","1hpv","1hsh","1hsi","1ivp","1sdt","2hb3","2hb4","2hpe","2hpf

type = c("PR1","PR1","PR2","PR1","PR1","PR2","PR2","PR2","PR1","PR1","PR1","PR2","PR2","PR1","PR2","PR1
names(type) = Proteases

pos.muT <-c(3,4,7,10,13,14,16,19,20,22,31:37,39:43,55:58,60:62,66:69,71:73,75:77,82,85,89,92:93,95,96,9
res.muT <-c(paste(pos.muT,"_A",sep=""), paste(pos.muT,"_B",sep=""))
```

Protocole

- step 1 : Détermination des classes d'atomes équivalentes chez PR1 et PR2
- step 2 : Classification des atomes des ligands de PR1
- step 3 : Classification des atomes des ligands de PR2
- step 4 : Pour chaque classe calculer la distance entre les atomes et le barycentre du superligand
- step 5 : Comparaison des distances moyennes entre les classes d'atomes équivalentes de PR1 et PR2

Calcul de la distance des atomes des ligands au barycentre du Superligand (SL)

les données

- dans le répertoire `script` : FileBasics.py, Geo3DUtils.py, PDB6.py : parseur des fichiers PDB
- dans le répertoire `data` : superligand.pdb : fichier PDB du superligand

Calcul des distances

```
#*****************************#
#           IMPORT            #
#*****************************#
import sys
import os
import string
```

```
from numpy import *
from math import sqrt
#*******************************#
#            PATH               #
#*******************************#
pathLig="data/ligands_pdb_cleaned_file_ATOM/"
pathsrc = "script/"
pathRes = "results/distance_LigAtom_SL/"
superligPDB = "data/superligand.pdb"
#*******************************#
#        IMPORT  PERSO          #
#*******************************#
sys.path.append(pathsrc)
from PDB6 import *
#*******************************#
# open file: pdb file of ligand#
#*******************************#
listTMP = os.listdir(pathLig)
listpdb = []
for elt in listTMP:
    if elt[-4:] ==".pdb":
        listpdb.append(elt)
#*******************************#
#         functions             #
#*******************************#
def getNmRes(resLine):
    num = resLine.rNum()
    ch = resLine.chnLbl()
    numRes = num + "_" + ch
    return(numRes)
def getDist(coord1,coord2):
    sumVal = 0
    for i in range(len(coord1)):
        sumVal = sumVal + ((coord1[i] - coord2[i])**2)
    dist = sqrt(sumVal)
    return(dist)

#*******************************#
#            main               #
#*******************************#
#Step 1 : computation of the superligand barycenter
pdb_obj = PDB(superligPDB)
listCoord = pdb_obj.xyz()
listCoord2 = [[],[],[]]
for si in listCoord:
    coord = si.split()
    coordNum = [float(i) for i in coord]
    listCoord2[0].append(coordNum[0])
    listCoord2[1].append(coordNum[1])
    listCoord2[2].append(coordNum[2])
coordBarySL = [mean(i) for i in listCoord2]
#Step 2 : calcul la distance entre les atomes d'une poche et le barycentre du SL
#Fichier
```

```
fileoutNm = "dist_AtomPocket_superlig.res"
fileout = open(pathRes+"/"+fileoutNm,"w")
for pdb in listpdb:
    pdbfile = os.path.join(pathLig,pdb)
    pdb_obj = PDB(pdbfile)
    for resLine in pdb_obj:
        numRes = getNmRes(resLine)
        for atmLine in resLine:
            atmNm = numRes + "_" + atmLine.atmName()
            coordAt = list(atmLine.xyz())
            distAtSL = getDist(coordBarySL, coordAt)
            ph = pdb + " " + atmNm + " " + str(distAtSL)
            fileout.write(ph+"\n")
fileout.close()
```

Ce programme a permis de générer le fichier `results/distance_LigAtom_SL/dist_AtomPocket_superlig.res` qui contient pour tous les ligands de PR1 et PR2 leur distance au SL.

Classification des atomes

Voici le code que j'avais fait précédemment pour classer les atomes des ligands chez toutes les PR2 disponibles dans la PDB

- étape 1 : création d'une matrice qui contient :
  - en lignes : les atomes de tous les ligands
  - en colonnes les coordonnées X, Y et Z (+ autre info des fichiers PDB).
    Les fichiers des ligands se trouvent dans cet exemple dans le répertoire PDB/Lig/

```
matAllLig = NULL
for (i in dir("PDB/Lig/")){
  codePDB = unlist(strsplit(i,"_"))[1]
  fileLig = read.table(paste("PDB/Lig/",i,sep=""))
  matAdd = data.frame(fileLig, rep(codePDB, length=nrow(fileLig)))
  matAllLig = rbind(matAllLig,matAdd )
}
```

- étape 2 : Classification des atomes des ligands.
  - calcul la distance Euclidienne entre tous les ligands à partir de leurs coordonnées 3D
  - calcul la classification hiérarchique
  - représente la classification hiérarchique

```
coord.At = matAllLig[,7:9]
hc = hclust(dist(coord.At), method="average")
plot(hc)
```

- étape 3 : Cherche le meilleur seuil de distance pour couper l'arbre pour créer les groupes d'atomes. On veut faire des groupes qui ne contiennent pas deux atomes extrait du même ligand Pour les différents seuils :
  - extrait les groupes
  - compte combien de protéines ont des atomes dans le même cluster. Le meilleur seuil est le plus grand seuil pour lequel il y a 0 protéine qui a au moins deux atomes dans un même ligand

```
for (HS in seq(0.8,1,by=0.005)){
  groupeAt = cutree(hc, h=HS)
```

```
  matAllLig.tmp = data.frame(matAllLig, groupeAt)
  tc = table(matAllLig.tmp[,13], matAllLig.tmp[,14])
  print(c(HS, nrow(which(tc >1, arr.ind=T))))
}
```

- étape 4 : Détermine les groupes avec le seuil choisi

```
seuil = 0.925
groupeAt = cutree(hc, h=seuil)
length(unique(groupeAt))
```

- étape 5 : Visulatisation de la taille des groupes

```
barplot(sort(table(groupeAt)))
hist(table(groupeAt), xlab="taille des clusters")
```