HMSN304 Structures des molécules
–

Projet Structure des molécules

Mouhamadou WADE
Jean MARONE

Enseignant:
Stefano TRAPANI

Matser 2 Science et Numérique pour la Santé
Parcours BCD
2019

Contents

1 Introduction **1**

2 Projet **2**

2.1 Projet (1/4) 2

2.1.1 Script 2

2.1.2 Résultat 3

2.2 Projet (2/4) 4

2.2.1 Script 4

2.2.2 Résultat du test 7

2.3 Projet (3/4) 10

2.4 Projet (4/4) 10

3 Conclusion **12**

1. Introduction

Le but de ce projet est de bien saisir le concept d'alignement de séquences protéiques sur des structures 3D connues. En effet, pour une séquence protéique donnée, dont la fonction et la structure sont inconnues, il faudra déterminer après une recherche et comparaison de séquences homologues dans des bases de données, où se trouve, dans la structure, le segment de séquence aligné. À quel résidu correspond le résidu n de la structure dans la séquence requête. Où se situent, dans la structure, les résidus conservés, et enfin quel est le rôle des résidus conservés. Pour cela , il faudra:

1. Attribuer une propriété atomique numérotant les résidus d'une structure.
2. Définir une propriété atomique pour comparer la séquence de résidus d'une structure avec celle d'une autre séquence après alignement.
3. Définir une procédure pour superposer deux structures homologues à la séquence requête.
4. Utiliser les procédures développées précédemment pour analyser les structures homologues de la protéine requête xy007.

La version 14.6.4 de Jmol a été utilisé pour réaliser ce projet.

2. Projet

2.1 Projet (1/4)

2.1.1 Script

Objectif:

L'objectif de la partie 1 du projet consiste à développer une *nouvelle propriété* atomique `property_seqresno` qui va donner une nouvelle numérotation séquentielle aux résidus d'un fichier PDB en suivant leur ordre d'apparition dans les enregistrement SEQRES. Les enregistrements SEQRES donnent l'ensemble des résidus de la protéine y compris ceux qui ne sont pas présents au niveau de la structure, mais aussi les résidus qui sont modifiés.

Pour définir cette nouvelle propriété `property_seqresno` , nous allons nous focaliser surtout sur la section COORDINATE[1] du fichier PDB. Pour rappel , un fichier PDB contient plusieurs *Sections*. Les plus importants pour nous seront :

- La section PRIMARY STRUCTURE qui contient la séquence de résidus dans chaque chaîne de macromolécule (s). Ces identifiants de chaîne et numéros de séquence permettent à d'autres enregistrements de se lier à la séquence. c'est dans cette section qu'on trouvera les enregistrements SEQRES.
- La section COORDINATE qui contient la collection de coordonnées atomiques ainsi que les enregistrements ATOM. C'est dans cette section qu'on trouvera les numéros de séquence des résidus (col 23-27) qu'il faudra changer avec la nouvelle propriété `property_seqresno`.

Procédé:

On définit d'abord quelques fonctions dont on aura besoin:

- `toUpperCase()` pour changer la casse en majuscule des paramètres d'entrée.
- `getFileHeader()` pour avoir l'entête du fichier PDB correspondant au modèle.
- `checkModID()` pour tester la validité de l'identifiant d'un modèle.

[1] https://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html

– `splitString()` pour couper une chaîne de caractères en morceaux de longueur fixe.

– `trimString()` qui enlève d'une chaîne un caractère donné en entrée et renvoie la chaîne sans le caractère spécifié.

– `getAllSeqFromSEQRES()` pour obtenir un tableau de l'ensemble des résidus d'un modèle donné.

Et puis on définit notre fonction `setPropertySeqresno(modID)`.
Pour obtenir les numéros de séquence des résidus d'un modèle donné, on définit une variable de type `bitset` (liste des numéros correspondant à la numérotation des atomes interne à Jmol), et chacun de ses atomes renvoie, par un numéro,au résidu et à la chaîne, auxquels il appartient.

On commence par vérifier la validité du modèle donné en entrée. Ensuite, on stocke le tableau contenant tous les résidus obtenue avec `getAllSeqFromSEQRES()` et le bitset du modèle sans le solvants ni le ligand au cas échéant.
Le bitset ainsi que le tableau sont parcouru en vérifiant pour chaque atome la chaîne et le résidu auquel il appartient. Pour chaque nouvelle chaîne du bitset, on réinitialise un compteur pour la numérotation.
La numérotation et l'attribution de la nouvelle propriété `property_seqresno` est effectuée au fur et à mesure du parcours du tableau et les résidus manquants sont ainsi gérés en se basant sur le tableau de résidus à chaque fois qu'un résidu du tableau est différent de celui du bitset.
Enfin, en parcourant simultanément de la sorte le tableau et le bitset, on a pas besoin de connaître les résidus modifiés pour les traiter. Il suffit juste de comparer si les résidus modifiés – s'il y en a – du tableau et ceux du bitset sont les mêmes.

2.1.2 Résultat

On a utilisé le fichier 1a2c.pdb pour effectuer les tests. Ce fichier contient 4 chaînes L,H,I et J.
On charge d'abord notre script ainsi que le fichier PDB de 1a2c.pdb :

```
1 $ script jmolGeneric.jmol ;
2 $ load 1a2c.pdb ;
```

Ensuite on définit notre fonction sur un modèle , puis on choisit une chaîne (ici chaîne L) pour illustrer.

```
1 /* On définit la nouvelle propriété sur un modèle */
2 $ setPropertySeqresno("1.1") ;
3
4 /* On choisit la chaîne L par exemple pour la démonstration */
5 $ chainL = {model = "1.1" and chain="L" and atomname="ca"};
6 $ chainL_seqcode = chainL.seqCode;
7
```

```
/* La numérotation de la chaîne L sans property_seqresno */
$ show chainL_seqcode;
$ chainL_seqcode = [ "1^H","1^G","1^F","1^E","1^D","1^C","1^B","1^A
    ","1","2","3","4","5","6","7","8","9","10","11","12","13","14","14^A
    ","14^B","14^C","14^D","14^E","14^F","14^G","14^H","14^I","14^J","14^K
    ","14^L","14^M","15" ]

/* Tester le premier élément de la chaine L */
$ print chainL[1].seqCode ;
$ 1^H
$ print chainL[1].property_seqresno ;
$ 1.0

/* Tester le dernier  élément de la chaine L */
$ print chainL.size ;
$ 36
$ print chainL[chainL.size].seqCode ;
$ 15
$ print chainL[chainL.size].property_seqresno ;
$ 36.0
```

Pour voir le résultat graphiquement au niveau de la structure , on a défini deux fonctions (`showSeqcode()`et `showSeqresno()` qui permettent de faciliter la visualisation.

```
/* Visualisation de la structure sans la propriété seqresno*/
$ showSeqcode("1.1");

/* Visualisation de la structure avec la propriété seqresno*/
$ showSeqresno("1.1");
```

On peut voir sur la Figure 2.1 la numérotation telle qu'elle est avant la définition de la propriété `property_seqresno` notamment pour la chaîne L (en jaune) et la chaîne I (en rouge).
La Figure 2.2 montre la numérotation après la définition de la propriété `property_seqresno` et on peut voir, comparé à précédemment que celle-ci recommence à 1 pour chacune des chaînes sauf pour la chaine "I" (en rouge) qui commence à 3 car les deux premiers résidus sont manquants.

2.2 Projet (2/4)

2.2.1 Script

Objectif:

Dans cette partie 2 du projet, nous allons définir deux propriétés atomique pour chaque résidu de la structure en se basant sur la numérotation obtenue avec la première fonction `setPropertySeqresno`.

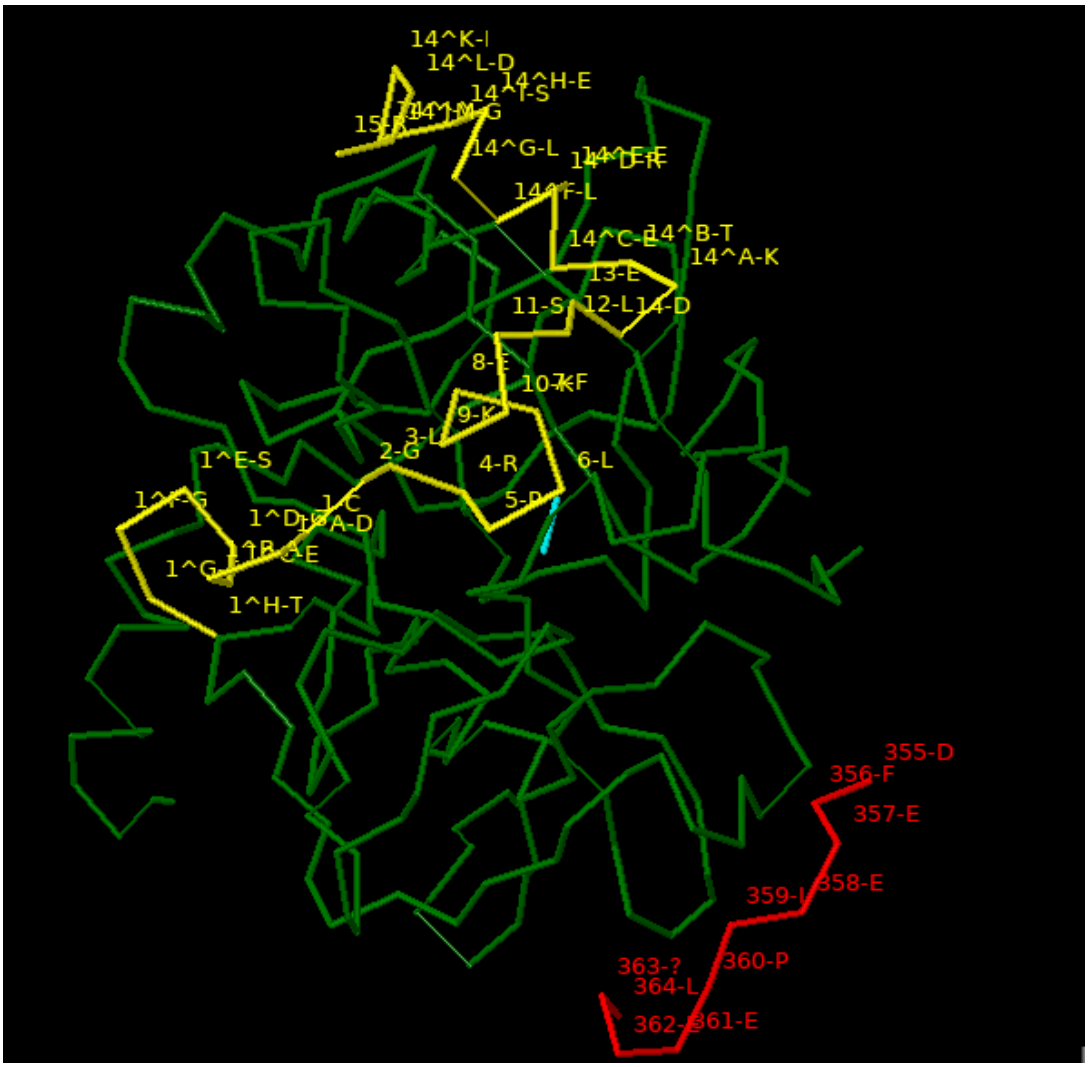

Figure 2.1: Numérotation des chaînes L(en jaune) et I(en rouge)
sans la propriété property_seqresno, obtenue avec `showSeqcode()`

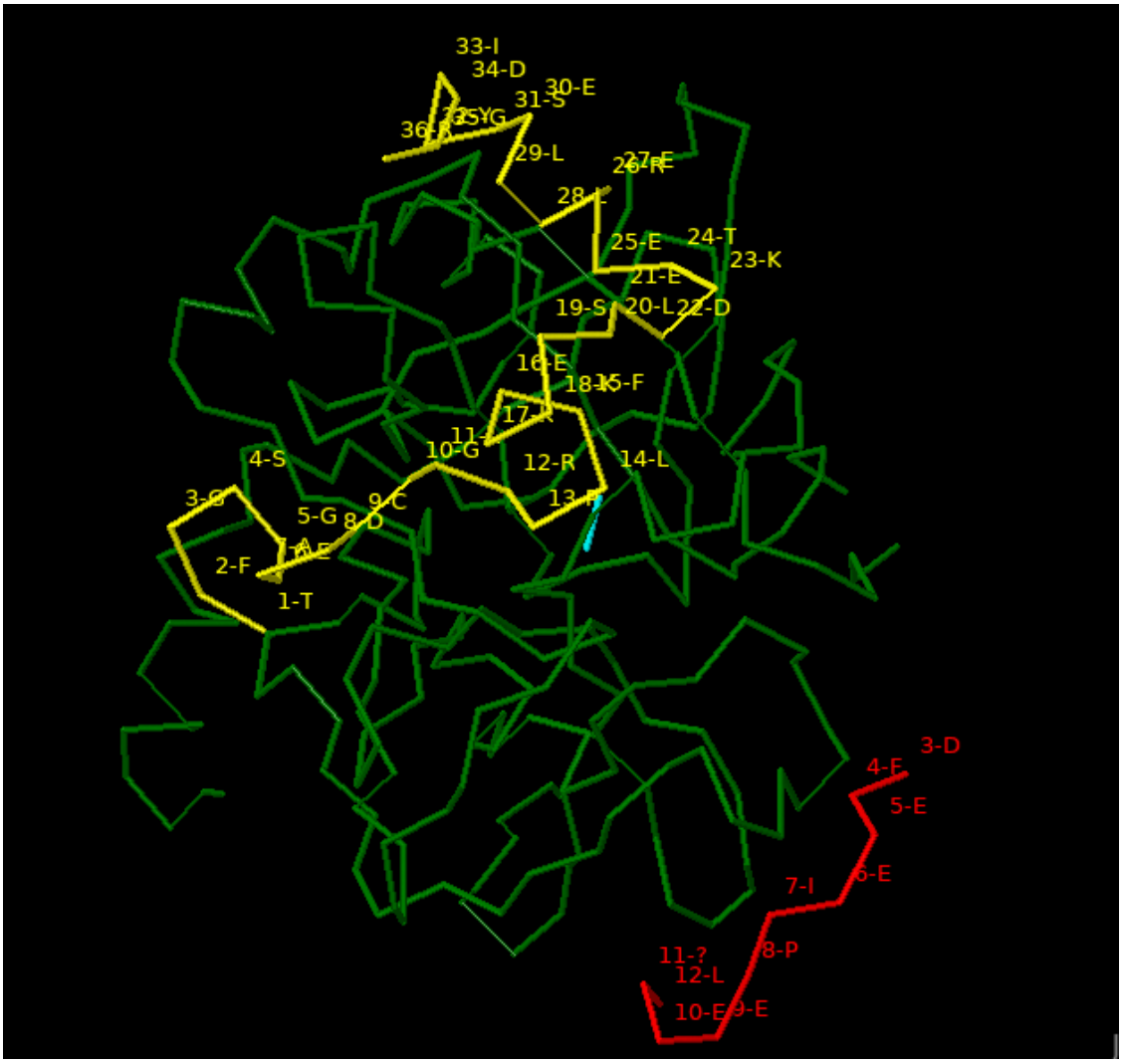

Figure 2.2: Numérotation des chaînes L(en jaune) et I(en rouge)
avec la propriété property_seqresno, obtenue avec `showSeqresno()`

La première, `property_qresno`,va correspondre au numéro du résidu de la séquence requête avec lequel le résidu a été aligné. On définit :

- `property_qresno` = 0 si le résidu est aligné avec un indel (gap)
- `property_qresno` = -1 si le résidu ne fais pas partie du segment aligné

La deuxième, `property_qsimil`, va définir la qualité de l'alignement à savoir :

- `property_qsimil` = -1 → résidu non aligné
- `property_qsimil` = 0 → résidu aligné dissimilaire
- `property_qsimil` = 1 → résidu aligné similaire
- `property_qsimil` = 2 → résidu aligné identique

Pour l'application de ces propriétés, on a à notre disposition des résultats d'alignements BLAST d'une séquence requête contre une structure protéique. Nous avons aussi les fichiers pdb de ces structures protéiques qui vont être utilisés au fur et à mesure dans notre fonction `setPropertyQresno`.

Procédé:

En plus des fonctions qu'on a eu à définir pour `setPropertySeqresno`, on va définir une nouvelle fonction :

- `getAllModel()` pour récupérer les numéros des modèles de notre fichier.

On commence par définir notre fonction `setPropertyQresno(blast_file)` qui prend en paramètre le résultat d'alignement BLAST. Puis on récupère les variables du fichier blast à savoir les numéros des zones d'alignement de la séquence requête et du Hit et les séquences de ces derniers. La fonction `getAllModel()` est alors utilisé pour récupérer les différents modèles existant dans notre fichier. On applique alors la fonction `setPropertySeqresno()`.
On initialise nos iterateurs, et pour tous résidus non alignés à la séquence requête, on fixe `property_qresno` à -1 et `property_qsimil` à -1 et pour résidu aligné on fixe `property_qsimil` à 1 si le résidu est similaire, `property_qsimil` à 2 si le résidu est identique, `property_qsimil` à 0 si le résidu est dissimilaire puis quand le résidu est dissimilaire en s'alignant avec un gap, `property_qresno` est fixé à 0.

2.2.2 Résultat du test

Pour le test de notre deuxième partie, on a eu à utiliser deux fichiers à savoir le résultat d'alignement entre une séquence requête et la protéine 2kfu contenue dans le fichier `BLAST_alignement1_2KFU-A.jmol` et le fichier 2kfu.pdb.
On commence d'abord par charger le script puis le fichier 2kfu.pdb

```
$ script jmolGeneric.jmol ;
$ load 2kfu.pdb ;
```

Ensuite on définit notre fonction `setPropertyQresno()` sur le fichier blast ce qui va appliquer la fonction `setPropertySeqresno` sur chaque modèle puis ajouter les propriétés `property_qsimil` et `property_qresno`.
Pour mieux visualiser les résultats obtenus, on va utiliser la représentation Backbone.

```
$ setPropertyQresno("BLAST_alignement1_2KFU-A.jmol");
$ select all;
$ backbone -0.3;
```

On va mettre en exergue le résultat en représentant les résidus non alignés `property_qsimil = -1` et les résidus alignés identiques `property_qsimil = 2` sans `setPropertySeqresno` à la figure 2.3 et avec `setPropertySeqresno` à la figure 2.4.

```
/* Script pour la représentation sans seqresno} */
$ select {atomname="CA" and property_qsimil=-1}; color red;
$ select {atomname="CA" and property_qsimil=2}; color cyan;

/* Script pour la représentation avec seqresno */
$ select {atomname="CA" and property_qsimil=-1};
$ color red;
$ label %.0[property_seqresno];
$ select {atomname="CA" and property_qsimil=2};
$ color cyan;
$ label %.0[property_seqresno];
```

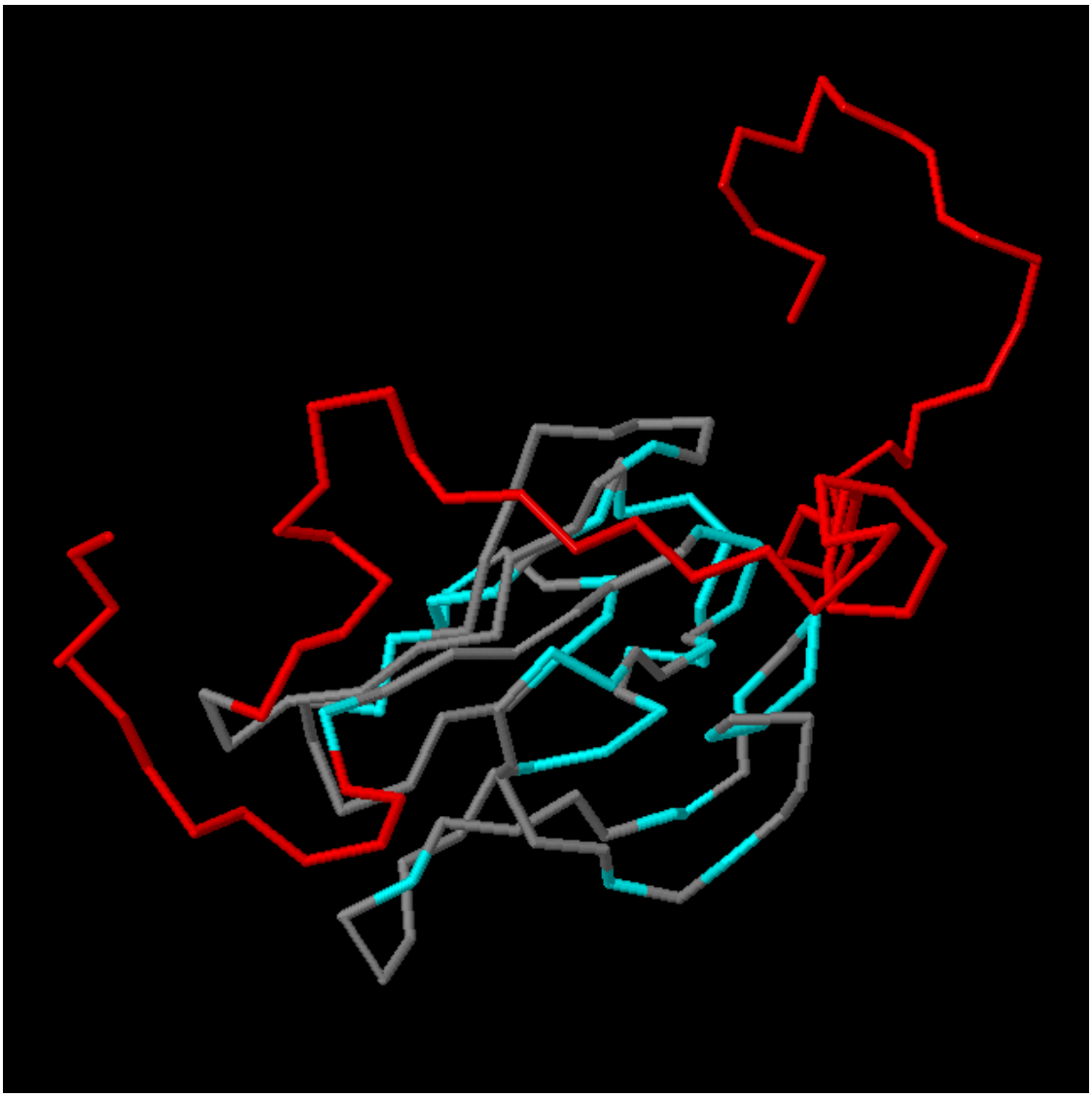

Figure 2.3: Représentation des résidus non alignés (en rouge) et des résidus alignés identiques (en cyan)

2.3 Projet (3/4)

Dans cette partie du projet, nous devons développer une procédure de superposition de deux structures homologues à une séquence requête. C'est à dire dans notre cas , on aura l'alignement entre une structure A et la séquence xy007 et l'alignement entre une structure B et xy007.
Cette superposition de structures doit se baser sur la définition préalable des paires d'atomes (Ai,Bi) à superposer (chaque paire d'atomes étant composé par un atome de la structure A et un atome de la structure B).
Le but de la superposition est d'emmener tous les atomes Ai le plus proche possible de leurs correspondants Bi en bougeant une des deux molécule par des rotations/translations sans déformer la structure interne des molécules. Afin de réaliser cette partie il faut tout d'abord choisir quels résidus nous voulons utiliser pour nos comparaisons avec les paramètres définis par `property_qsimil` dans la partie 2 du projet.
Donc pour superposer A et B, on construira les paires d'atomes (Ai,Bi) qu'on aura choisi de superposer en appariant Ai et Bi seulement si Ai et Bi sont alignés avec le même résidu de xy007.

Malheureusement, cette partie du projet n'a pas été effectuée, en tous cas jusqu'à son terme, car ayant eu un peu de difficultés déjà à comprendre et finaliser à temps les parties 1 et 2 du projet.

2.4 Projet (4/4)

Cette partie consiste à utiliser les procédures précédemment développées dans les 3 parties pour analyser les structures homologues de la protéines xy007.
On a pu réaliser une partie de cette quatrième étape dans la seconde partie du projet avec les propiétés `property_qsimil` et `property_qresno`.

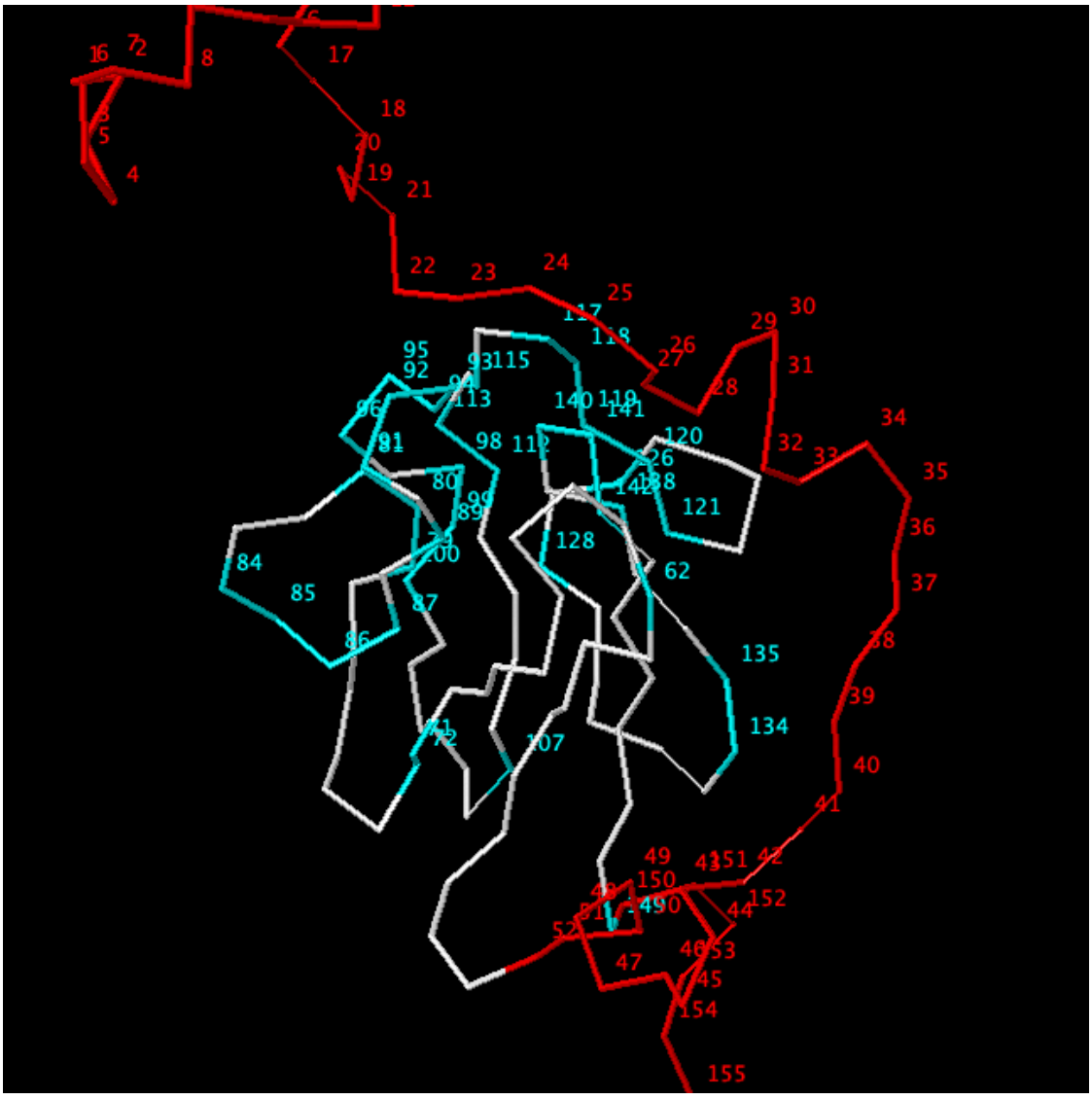

Figure 2.4: Représentation des résidus non alignés (en rouge) et des résidus alignés identiques (en cyan) avec la propriété seqresno

3. Conclusion

Ce projet de structure des molécules nous a permis de mieux comprendre la modélisation de molécules biochimiques ce qui est un apport important à notre formation. On s'est familiarisé plus aux structures des fichiers PDB, à leur architecture mais également aux sorties que nous fournissent les fichiers d'alignement BLAST. À travers le logiciel Jmol et son langage de script qui permet la création de procédures adaptées au besoins de l'utilisateur, on a pu réaliser ce projet.