Annotation des structures de la PDB en domaines et étude des domaines en interaction pour voir si certaines paires de domaines intéragissent plus que les autres

Ariane Odjo & Nourdine Bah
M2BIBS, projet ITPP

3 décembre 2014

Table des matières

Table des figures

1 Présentation du sujet

1.1 Notre interprétation du sujet

Suite à notre entrevue avec le Dr. Anne Lopes, au cours du dernier cours de programmation python, nous pensions avoir compris le sujet, et avons fait l'erreur de penser que nous pourrions le retrouver sans prendre de notes. Cependant nous nous sommes surrestimés et sommes partis sur quelque chose de totalement différent, à savoir évaluer la fréquence avec laquelle les domaines d'une protéine donnée interagissent avec les domaines d'autres protéines. Notre échange d'*emails* nous a heureusement permis de re-clarifier les choses, et nous avons décidé de repartir sur les consignes qui nous été données. Cela nous paraissait plus intéressant que ce que nous faisions initialement.

L'idée générale est donc d'évaluer la fréquence d'interaction pour chaque paire de domaines susceptibles d'interagir (paraphrase de l'*email* du Dr. Lopes).

1.2 Ce que nous avons essayé de faire

Les annotations de la PFAM donnent pour chaque structure PDB les différents domaines connus que la structure contient, et la localisation de leurs résidus. La définition de chaque domaine est issue de la base de donnée de PFAM. De là nous sommes partis sur un programme qui prend un domaine de la PFAM et calcule la fréquence d'interaction avec les autres domaines, et avons défini les différentes classes dans cet objectif. Cependant nous nous sommes rendus compte qu'il fallait aussi que le programme puissent simplement prendre en entrée une liste de fichier PDB, et nous avons rajouter des fonctions. Cela se ressent dans le code du programme car les fonctions ajoutées ne sont pas forcément cohérentes avec les classes, et les arguments des classes lors de leur création sont devenus inutiles dans certains cas[1]. Nous avons également constaté que la base de donnée iPFAM répertoriait des prédictions d'interaction entre les différents domaines de la PFAM, et avons décidé de comparer cela avec les résultats que nous obtenons.

Il y a donc trois cas de figures pour le programme :
— calcul de la fréquence d'interaction pour un domaine PFAM donné sans *a priori* ;
— calcul de la fréquence d'interaction pour un domaine PFAM donné avec consultation d'iPFAM ;
— calcul des fréquences d'interactions de tous les domaines d'une liste de fichier PDB.

2 Fonctionnement du programme

2.1 Classes du programme

Le programme comporte 6 classes :
— une classe `domain` qui s'occupe des domaines PFAM ;
— une classe `structure` qui regroupe les fonctions de gestion des fichiers de structure PDB ;
— une classe `interaction` qui regroupe les fonctions ayant à trait aux interactions entre les domaines ;
— une classe `graph` qui gère la représentation des fréquences sous forme de graphe. Elle fait appel au module `pydot` ;
— une classe `matrixcsv` qui gère la représentation des fréquences sous forme de matrice au format CSV ;
— une classe `representation` qui gère l'affichage des domaines et résidus en interaction via `pymol`.

1. La chaîne `"whatever"` fut passée en argument dans ces cas là.

2.2 Script principal

2.2.1 Fréquences d'interaction d'un domaine sans *a priori*

```
domain = odjo.domain(sys.argv[1])
pdb_ids = domain.get_pdb_ids()
inter = odjo.interaction(domain.pfam_id, pdb_ids, "whatever")
annot = inter.get_annotations()
analysis = inter.analysis_2(annot)
freq = inter.frequencies(analysis)
odjo.graph().draw(freq)
odjo.matrixcsv().frequencies_matrix(freq)
```

Listing 1 – Code pour des fréquences d'interaction d'un domaine PFAM sans *a priori*.

Voici l'explication du code :

1. La fonction `odjo.domain()` créé un objet `domain` à partir d'un identifiant PFAM ;
2. La fonction `domain.get_pbd_ids` retourne, sous forme de liste, les idenfifiants PDB des strucutures contenant le domaine PFAM ;
3. La fonction `odjo.interaction()` créé un objet `interaction`. Le premier argument est l'idenfiant du domaine, le second la liste des identifiant des structures le contenant et le dernier argument peut ne pas être valide ;
4. La fonction `get_annotations()` retourne une liste de dictionnaires contenant les annotations des domaines pour les fichiers PDB contenant le domaine d'intérêt. Le schéma de chaque dictionnaire est celui-ci :

```
    {
      'pfam_id':id,
      'pdbs': [
        {
          'pdb_id':id,
          'domains': [
            {'id': id_pfam, 'name': pfamName, 'chain':'A', 'start': 50, 'end':
60},
            ...,
            {},
            {}
          ]
        },
        ...,
        {}
      ]
    }
```

;

5. La fonction `analysis_2()` retourne un dictionnaire contenant le nom du domaine d'intérêt et la liste des domaines avec lesquels il interagit et chaque élément de la liste est une observation :

```
    {
      'name':pfam_name,
      'interactants': [
        interacting_domain_1,
        interacting_domain_2,
        interacting_domain_2,
        interacting_domain_1
        ...,
        interacting_domain_1,
        interacting_domain_3
```

```
        ]
    }
```

;

6. La fonction `frequencies()` calcule la fréquence des interactions observées, elle retourne une liste de dictionnaires de ce type :

```
    {
        'domain_1': domain_name,
        'domain_2': domain_name,
        'frequency': 0.08
    }

```

;

7. La fonction `odjo.graph().draw` exporte, au format PNG, un graphe où les sommets sont les domaines et où les arrêtes sont pondérées par les fréquences d'interaction ;
8. La fonction `odjo.matrixcsv().frequencies_matrix()` exporte, au format CSV, une matrice contenant les fréquences d'interaction entre les domaines.

2.2.2 Fréquences d'interaction d'un domaine avec iPFAM

```
domain = odjo.domain(sys.argv[1])
pdb_ids = domain.get_pdb_ids()
interacting_domains = domain.get_interactions()
inter = odjo.interaction(domain.pfam_id, pdb_ids, interacting_domains)
annot = inter.get_annotations()
analysis = inter.analysis(annot)
freq = inter.frequencies(analysis)
odjo.graph().draw(freq)
odjo.matrixcsv().frequencies_matrix(freq)
```

Listing 2 – Code pour des fréquences d'interaction d'un domaine PFAM avec iPFAM.

La première différence avec le code précédent est l'utilisation de la fonction `analysis()` au lieu de `analysis_2()`. Dans ce cas de figure le programme ne compare pas tous les domaines, mais seulement les domaines connus pour interagir avec le domaine d'intérêt. La seconde différence est que le dernier arguments de la fonction `odjo.interaction()` doit être valide.

2.2.3 Fréquences d'interaction avec une liste de fichiers PDB

```
pdb_ids = file2list(sys.argv[1])
struct = odjo.structure(pdb_ids[0])
annotations = struct.get_domains_from_list(pdb_ids)
inter = odjo.interaction("whatever", "whatever", "whatever")
analysis = inter.analysis_3(annotations)
freq = inter.frequencies_from_list(analysis)
odjo.graph().draw_list(freq)
odjo.matrixcsv().frequencies_matrix(freq)
```

Listing 3 – Code pour des fréquences d'interaction d'un domaine PFAM avec iPFAM.

Voici l'explication du code :

1. La fonction `file2list()` récupère le fichier et créé une liste ;

2. La fonction `odjo.structure` crée une classe `structure` (un argument valide n'est pas nécessaire) ;
3. La fonction `get_domains_from_list()` récupère tous les fichiers PDB de la liste ;
4. La fonction `odjo.interaction()` créé un objet `interaction` (des arguments valides ne sont pas nécessaires) ;
5. La fonction `analysis_3` est la même que la fonction `analysis_2` mais elle renvoie une liste de dictionnaires car plusieurs domaines ont été analysés ;
6. Les fonctions `frequencies()`, `odjo.graph().draw` et `odjo.matrixcsv().frequencies_matrix()` sont les mêmes que précédemment.

3 Résultat

Nous ne pouvions pas tester tous les domaines/familles de la PFAM, ni tous les fichiers PDB. Nous nous sommes donc limités à une famille de protéines dites de la famille de Bcl-2.

3.1 Comparaison entre iPFAM et notre programme

iPFAM est une programme élaboré et éprouvé : rien de comparable avec le notre ! iPFAM prédit les interactions entre résidus en calculant toutes les liaisons de van der Waals, les liaisons entre chaînes latérales, les liaisons hydrogènes, les ponts salins et les ponts disulfures entre les paires de résidus des différents domaines [Finn et al., 2005]. Voici le code que nous utilisons pour détecter les interactions :

```
atoms = Selection.unfold_entities(model, 'A')
nsearch = NeighborSearch(atoms)
interacting_atoms_1 = []
interacting_atoms_2 = []
for atom in atoms:
    if atom.get_serial_number() in numbers_1:
        point = atom.get_coord()
        neighbors = nsearch.search(point, 5)
        for neighbor in neighbors:
            if neighbor.get_serial_number() in numbers_2:
            interacting_atoms_2.append(neighbor)
                if atom not in interacting_atoms_1:
                    interacting_atoms_1.append(atom)
```

Listing 4 – Code de la fonction *interaction()*.

On peut voir que nous contentons de sélectionner les atomes situés au plus à 5 Å d'atomes d'un autre domaine. Nous avions écrit cette fonction en première intention, et avions prévu d'y revenir plus tard pour l'améliorer [2], mais nous n'avons pas eu le temps.

Nous avons néanmoins tester notre programme avec ou sans iPFAM pour l'identifiant `PF00452` qui correspond à un des domaines caractéristiques des protéines de la famille de Bcl-2. Le graphe réprésentant la recherche sans iPFAM correspond à la figure 1, et celui correspondant à la recherche avec iPFAM correspond à la figure 2. Nous avons également ajouté la matrice de fréquences pour la recherche sans iPFAM, c'est le tableau 1 (l'autre matrice était trop grosse).

Nous pouvons constater que notre programme sans iPFAM détecte 9 domaines de plus qu'avec iPFAM :

— Ank_1 ;
— Ank_5 ;
— Ank_2 ;

2. nous pensions notamment à une analyse des surfaces d'interaction

— SBP_bac_1 ;
— Ank_3 ;
— SBP_bac_8 ;
— Ank ;
— P53 ;
— SBP_bac_6.

Les domaines SBP_bac correpondent à des protéines bactériennes, et leur interaction avec les protéines de la famille de Bcl-2 ne semble pas avoir de signification biologique. Cependant, l'interaction avec P53 est très bien décrite et est très importante [Ha et al., 2013]. L'interaction avec P53 peut également expliquer le résultat pour les répétitions ankyrines via 53BP2 [Gorina and Pavletich, 1996].

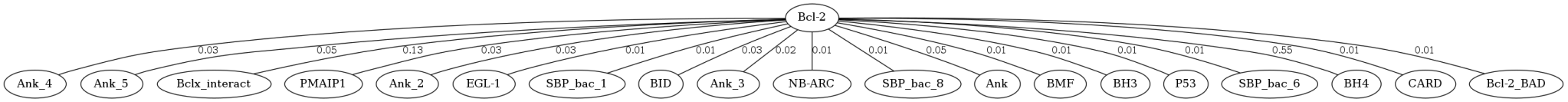

FIGURE 1 – Graphe pour *PF00042* sans iPFAM.

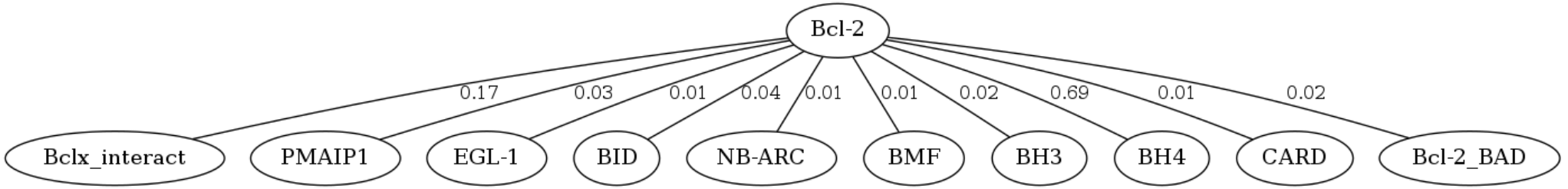

FIGURE 2 – Graphe pour *PF00042* avec iPFAM.

	Bcl-2	Bclx_interact	PMAIP1	EGL-1	BID	NB-ARC	BMF	BH3	BH4	CARD	Bcl-2_BAD
Bcl-2	0	0.17	0.03	0.01	0.04	0.01	0.01	0.02	0.69	0.01	0.02
Bclx_interact	0	0	0	0	0	0	0	0	0	0	0
PMAIP1	0	0	0	0	0	0	0	0	0	0	0
EGL-1	0	0	0	0	0	0	0	0	0	0	0
BID	0	0	0	0	0	0	0	0	0	0	0
NB-ARC	0	0	0	0	0	0	0	0	0	0	0
BMF	0	0	0	0	0	0	0	0	0	0	0
BH3	0	0	0	0	0	0	0	0	0	0	0
BH4	0	0	0	0	0	0	0	0	0	0	0
CARD	0	0	0	0	0	0	0	0	0	0	0
Bcl-2_BAD	0	0	0	0	0	0	0	0	0	0	0

TABLE 1 – Matrice de fréquence pour *PF00042* avec iPFAM.

3.2 Test sur tous les fichiers PDB des protéines de la famille de Bcl-2

Le graphe correpondant à cette recherche est représenté sur la figure 3. On y retrouve les interactions importantes pour le fonctionnement des protéines de la familles [Juin et al., 2013], avec notamment les domaines :

— BH3 ;
— BH4 ;
— BID ;
— BMF ;
— CARD.

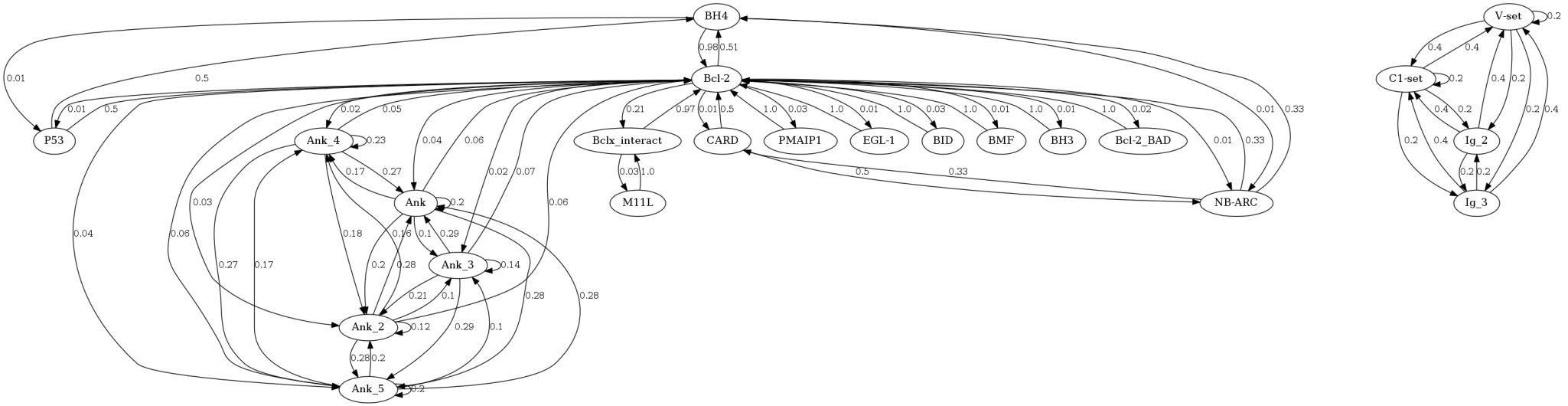

FIGURE 3 – Graphe pour les fichiers PDB des protéines de la famille de Bcl-2.

3.3 Problème pour l'affichage des structures

Nous avons tenté de représenter, sous la forme d'image, les structures annotées avec des couleurs codes, et avec les résidus interagissant en représentation surface. Cependant pymol s'arrête lorsqu'il y a plusieurs fichiers à exporter. Nous n'avons pas réussi à résoudre le problème, néanmoins la figure 4 est un exemple de ce que nous obtenons lorsqu'il n'y a qu'une interaction détéctée (et donc un seul fichier à exporter).

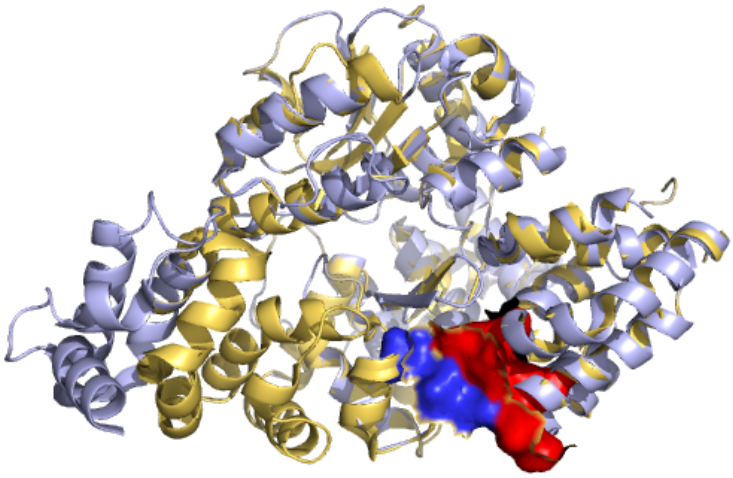

FIGURE 4 – Exemple d'export, au format PNG, d'une structure annotée.

4 Les défauts de notre programme et volonté d'amélioration

Le premier problème est la fonction qui détecte les interactions (distance de 5 Å). Elle est trop simpliste et doit tenir compte de ce qu'est réellement un interaction protéine-protéine.

Notre programme est mal organisé, et les classes ne sont pas forcément cohérente.

Notre programme ne tient pas compte des homo-interactions lorsqu'il prend en entrée un domaine PFAM.

La gestion des erreurs n'est pas bonne. Nous n'avons pas eu le temps d'ajouter des *exceptions*, et le programme peut s'arrêter par exemple lorsqu'il télécharge un fichier PDB qui n'existe pas.

Pour récupérer les annotations PFAM, les connections au serveur se font les unes après les autres pour chaque identifiant PDB. Cela prend du temps, mais nous n'avons pas trouvé un moyen pour récupérer toutes les annotations en une seule fois. De la même manière il serait plus pratique de récupérer toutes les structures PDB dans un seul et même fichier.

Nous avons sûrement oublié d'autres défauts.

Merci pour le temps passé à lire ce rapport.

Références

[Finn et al., 2005] Finn, R. D., Marshall, M., and Bateman, A. (2005). iPfam : visualization of protein-protein interactions in PDB at domain and amino acid resolutions. *Bioinformatics*, 21(3) :410–412.

[Gorina and Pavletich, 1996] Gorina, S. and Pavletich, N. P. (1996). Structure of the p53 tumor suppressor bound to the ankyrin and SH3 domains of 53BP2. *Science*, 274(5289) :1001–1005.

[Ha et al., 2013] Ha, J. H., Shin, J. S., Yoon, M. K., Lee, M. S., He, F., Bae, K. H., Yoon, H. S., Lee, C. K., Park, S. G., Muto, Y., and Chi, S. W. (2013). Dual-site interactions of p53 protein transactivation domain with anti-apoptotic Bcl-2 family proteins reveal a highly convergent mechanism of divergent p53 pathways. *J. Biol. Chem.*, 288(10) :7387–7398.

[Juin et al., 2013] Juin, P., Geneste, O., Gautier, F., Depil, S., and Campone, M. (2013). Decoding and unlocking the BCL-2 dependency of cancer cells. *Nat. Rev. Cancer*, 13(7) :455–465.