Lexique utilisé :
aa → acides aminés
Repeat → sous-unité d’un domaine bêta-propeller (= feature)
Blade → sous-unité d’un domaine bêta-propeller (= repeat)
Classe → désigne un ensemble de blades portant le même numéro de repeat (1, 2, 3 ...)

Cahier de laboratoire :
    Format :
    {**AAAA-MM-DD :**
    travail exécuté par : <"noms">
    "contenu"
    <figures/images>
    }

Nous travaillons sur les bêta-propellers. Ces protéines sont constituées d’éléments répétés qui s’agencent sous forme de donut composé de plusieurs blades. Chaque élément répété forme un blade et possède une extension qui interagit avec le blade précédent.

2024-10-19 :

Travail exécuté par : MOSER Mathilda

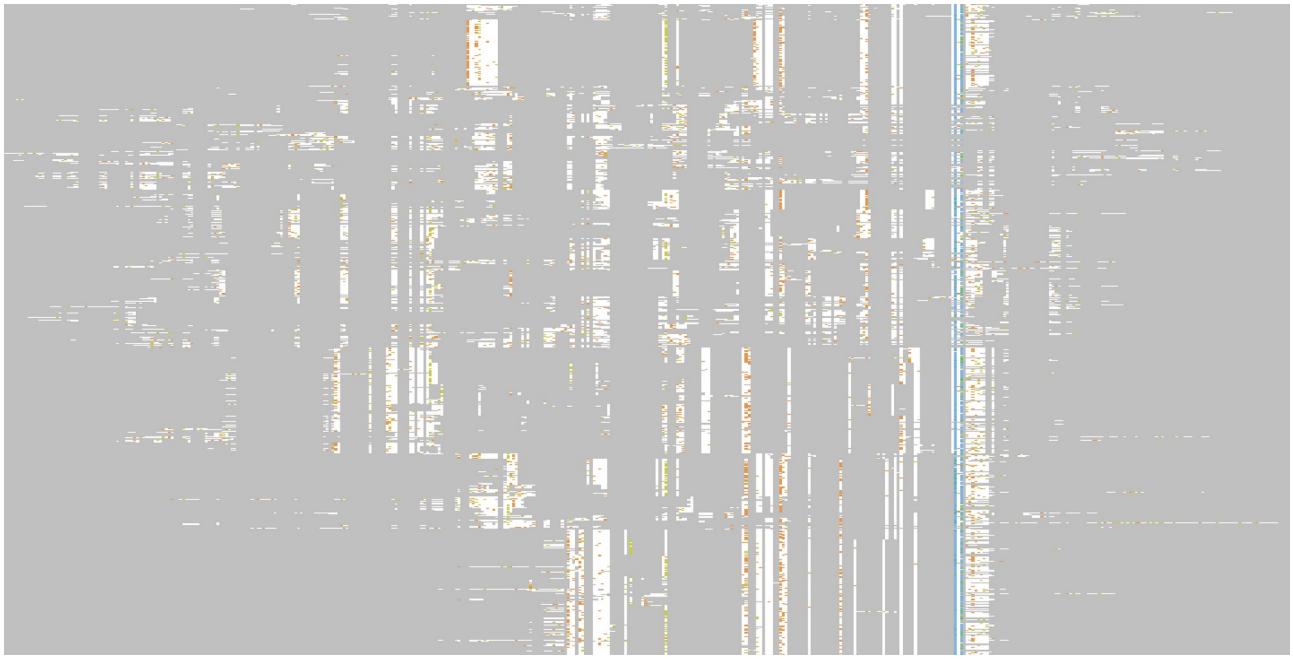
Production des premiers alignements à partir des séquences fasta des WD repeats, des WD repeats étendus de 10 aa de chaque coté et des WD repeats étendus de 20 aa de chaque coté. Les fichiers de sortie sont réorganisés pour rassembler les séquence alignées ayant un grand nombre de correspondance le long de leur séquence entre elles sous forme de blocs, ce qui facilite l’analyse de ces alignements.
Ces alignements ont été obtenus grâce à la version web de clustalΩ disponible à cette adresse : " https://www.ebi.ac.uk/jdispatcher/msa/clustalo " . Ce premier choix d’utiliser clustalΩ comme outil d’alignements multiples a été fait car c’est un outil qui permet en général de bien alignés des séquences courtes en aa comme celles que nous utilisons. Cependant, la grande variabilité des longueurs de ces séquences peuvent représenter un obstacle dans l’alignement avec cet outil. Les paramètres d'entrée ont été laissés par défaut sur le site sauf pour le format de sortie qui à été changé vers le format FASTA.

(PARAMETRES UTILISEES :
  - OUTPUT FORMAT : Pearson/FASTA
  - DEALIGN INPUT : NO
  - MBED-LIKE CLUSTERING GUIDE-TREE : yes
  - MBED-LIKE CLUSTERING ITERATION : yes
  - COMBINED ITERATIONS : default(0)
  - MAX GUIDE TREE : default
  - MAX HMM ITERATIONS : default
  - ORDER : aligned
  - DISTANCE MATRIX : no
  - OUTPUT GUIDE TREE : yes)

Les résultats ont été obtenus au format FASTA et ont été visualisés en utilisant le programme Jalview v2.11.4.0.

À cette étape du projet, aucune conclusion ni correction n'a été apporté vis-à-vis de ces alignements.

Figure 1 : Overview de l’alignement réalisé avec ClustalΩ



2024-10-29

Travail exécuté par : WITTENMEYER Guillaume

Pour comparer avec avec l'alignement de clustalΩ, nous avons décidé d’utiliser en complément la génération d'alignement de ces séquences grâce au programme MAFFT via l'interface web " https://usegalaxy.eu " avec l'historique accessible ici : " https://usegalaxy.eu/u/encelade/h/wdrepeats ". L’utilisation de cet outil s’explique du fait qu’il peut potentiellement mieux prédire les alignements multiple pour des séquences qui divergent beaucoup comme nos séquences de WD repeats qui sont très variable en longueurs.

(PARAMETRES UTILISES :
 - Type of sequences : Amino acids
 - Type of scoring matrix : BLOSUM
 - Coefficient of the BLOSUM matrix : 62
 - Configure gap costs : Use default values
 - Reorder output : Yes/No (générer un de chaque)
 - Output format : FASTA)

Les résultats ont été enregistrés au format FASTA avec le PATH: " ~/common/data/1_intermediate/MAFFT_alignment "
Les résultats ont été obtenus au format FASTA et ont été visualisés en utilisant le programme Jalview v2.11.4.0.

2024-10-29

Travail exécuté par : WITTENMEYER Guillaume, MOSER Mathilda, KESHAVARZ-NAJAFI Mohsen

Production de nouveaux alignements en utilisant MAFFT. Nous avons généré, pour chaque fichier d'entrée, deux fichiers de sorties. Le premier fichier de sortie possèdent l'ordre du fichier d'entrée. L'ordre des séquences a été réorganisé automatiquement pour le deuxième fichier.
Ces étapes ont été réalisées à partir des séquences fasta des WD repeats, des WD repeats étendus de 10 aa de chaque coté et des WD repeats étendus de 20 aa de chaque coté.
Nous avons donc par la suite choisi de ne travailler qu'avec les alignements produits par l'outil MAFFT car ces derniers nous ont semblé mettre en évidence plus de positions conservés ou de propriétés physico-chimiques proches.

Les résultats ont été obtenus au format FASTA et ont été visualisés en utilisant le programme Jalview v2.11.4.0.

2024-10-29

Travail exécuté par : WITTENMEYER Guillaume, MOSER Mathilda, KESHAVARZ-NAJAFI Mohsen

A partir des premiers alignements réalisés avec MAFFT, nous avons pu extraire plusieurs positions intéressantes en nous basant dans un premier temps sur les alignements des séquences sans extensions (~/common/data/1_intermediate/MAFFT_alignment).
Tous d'abord nous remarquons la présence de beaucoup de positions à tendance apolaire (généralement des aa aliphatiques et parfois aromatiques). Les motifs WD-repeat sont généralement caractérisés par un dipeptide GH en début de séquence et d'un dipeptide WD en fin de séquence. Ces deux motifs semblent être présent dans tous nos alignements, et c'est ce que nous souhaitons vérifier.

Le tableau suivant récapitule toute les positions qui ont été trouvées. L'analyse a été faite par visualisation de l'alignement obtenu avec MAFFT du 2024-10-29 (format FATSA) grâce à l'application Jalview v2.11.4.1 (mais aussi avec la v.2.11.4.0). Les nombres après chaque aa indique la proportion de l'acide aminé à cette position (en pourcentage) et sont issus de Jalview.

Figure 2 : tableau récapitulatifs des positions sélectionnées.

Positions	aa dominants	Propriétés physico-chimique
71	G (23,36)	/
72	H (42, 19)	/
128	V (39,37), I (22,68), L (9,76), A (8,96)	Position hydrophobe (Aliphatique)
161	L (27,08), V (23,93), I (12,58)	Position hydrophobe (Aliphatique)
204	F (28,11), W (18,58), L (9,39), V (6,38),	Position hydrophobe (Aliphatique, Aromatique)

	Y (5,96), I (4,41)	
215	S (25,11)	/
255	P (36,32)	/
300	G (27,92)	/
339	L (39,84), I (16,28), V (13,89), F (8,59)	Position hydrophobe (Aliphatique, Aromatique)
359	A (26,54), V (21,07), L (16,49), I (8,17), F (5,58)	Position hydrophobe (Aliphatique, Aromatique)
374	S (30,22), T (24,96), A (12,29)	Petite molécule à tendance polaire non ionisable
386	G (40,78), A (16,61)	Petite molécule apolaire
416	S (32,00), G (15,95)	/
444	D (52,84)	Position acide
469	G (33,88)	Petite molécule
482	T (20,51), S (12,39)	Petite molécule à tendance apolaire non ionisable
491	V (32,80), I (25,95), L (20,27)	Position hydrophobe
509	R (20,60), K (17,46)	Position basique
528	L (26,04), V (23,60), I (22,99)	Position hydrophobe (Aliphatique)
560	W (56,36), Y (11,45), F (10,00)	Position hydrophobe (Aromatique)
572	D (37,35), N (11,12)	Position polaire à tendance ionisable
588	L (20,69), V (13,23), I (8,22)	Position hydrophobe (Aliphatique)

2024-11-19

Travail exécuté par : WITTENMEYER Guillaume

Suite à nos précédentes observations, pour améliorer nos alignements, nous avons pensé qu'il serait intéressant d'aligner chaque classe de repeats en utilisant MAFFT indépendamment les unes des autres pour voir s’il n'existe pas de features intéressantes à mettre en évidence pour chaque classe de repeats.
Suite à nos réunions avec le Groupe 3, nous avons aussi jugé qu'il était intéressant de de ne garder que les protéines ayant un nombre de repeats maximum de 7 (donc 1 seul domaine) pour faciliter les analyses (voir https://github.com/Hudego/PTU_Project_3.git). Ce choix à été fait puisque la majorité des protéines ont 7 repeats. On peut également supposer que les protéines possédant un nombre de repeats égal à 7 sont plus proches entre elles

qu’avec des protéines ayant un nombre de repeats variable. Il est donc préférable de conserver uniquement les domaines qui possèdent le même nombre de repeats. Pour rester cohérent avec notre étude, nous avons décidé de ne nous baser que sur les protéines de nos alignements ayant un maximum de 1 domaine de 7 repeats. Cela permettra également potentiellement de réduire le bruit introduit par les domaines particuliers à nombre de repeats variables qui sont possiblement pus éloignés.
Pour effectuer cela nous avons écrit et utilisé un script python (~/common/scripts/filter.py) dans un environnement conda spécifique au projet en python=3.9.20 (~/common/scripts/wd_repeat.yml) dont les dépendances et librairies sont spécifiées dans le fichier README.md du projet.
Les fichiers fasta filtrés générés ont été sauvegardés dans des sous-dossiers séparés dans le dossier ~/common/data/0_raw/sequences_fasta/

Alignement de ces séquences grâce au programme MAFFT via l'interface web " https://usegalaxy.eu " avec l'historique accessible ici : " https://usegalaxy.eu/u/encelade/h/wdrepeats "

(PARAMETRES UTILISES :
- Type of sequences : Amino acids
- Type of scoring matrix : BLOSUM
- Coefficient of the BLOSUM matrix : 62
- Configure gap costs : Use default values
- Reorder output : Yes
- Output format : FASTA)

Les résultats ont été enregistrés au format FASTA avec le PATH: " /data/projet5/common/data/1_intermediate/MAFFT_alignment/" dans un sous-dossier séparé pour chaque raw data.
Les résultats ont été obtenus au format FASTA et ont été visualisé en utilisant le programme Jalview v2.11.4.0.

2024-11-20

Travail exécuté par : MOSER Mathilda

Avec les alignements de la veille (réalisés par Guillaume), nous avons réalisés des weblogos pour essayer de trouver des positions intéressantes. Les weblogos sont disponibles au format PDF à ce PATH:" ~/common/data/1_intermediate/weblogo/weblogo_full ". Sur ces weblogos, nous retrouvons bien les motifs GH en débuts de séquences (les GH en fins de séquences qui apparaissent sur les weblogos des séquences avec des extensions représentent les GH de début de séquences du repeat suivant) et WD. Nous constatons aussi la présence de beaucoup de positions hydrophobes. Les positions hydrophobes sont en noires.

Les weblogos ont étés générés grâce à l’outil en ligne Seq2Logo (https://services.healthtech.dtu.dk/services/Seq2Logo-2.0/).

2024-11-21

Travail exécuté par : KESHAVARZ-NAJAFI Mohsen

Suite à ces observation, nous avons décidé de manuellement effectuer un fichier excel qui recense toutes les observations de conservation pour ces séquences (~/common/data/1_intermediate/excels/Conservation_classfiltered_alignment.xlsx).

Dans notre fichier Excel, nous avons essayé de noter toutes les conservations supérieures à 10 % au vu de la qualité globale médiocre de ces alignements observées dans les 16 fichiers d'alignements MAFFT, visualisés dans Jalview, pour les séquences WD et WD-10. Les alignements avec 20 aa d'extensions perturbant les alignements, il n'y a que très peu de positions conservées à plus de 10 % dans ces alignements. Nous avons également marqué en couleurs les conservations ayant un pourcentage plus élevé ou celles qui se répétaient dans différents fichiers, afin d'avoir une vue d'ensemble plus claire.
Selon nos observations, en plus des conservations dans les régions WD, une conservation H a été identifiée dans les parties initiales de chaque alignement. Dans les fichiers contenant 10 résidus d'extension, cette conservation H se répète également dans la partie finale ce qui correspond au début de la séquence du repeat suivant. De plus, juste avant chaque H, on peut observer une conservation G. Au milieu de tous les alignements, il y a également une conservation P qui pourrait appartenir à une boucle de liaison dans le repeat. Une autre conservation commune dans tous les alignements est celle des acides aminés (V, I, L), retrouvée dans différentes régions. On peut supposer que ces trois acides aminés, en raison de leurs propriétés physico-chimiques proches, se remplacent parfois entre eux. Cependant, leurs caractéristiques fondamentales d'être hydrophobes et non polaires sont préservées, ce qui pourrait être essentiel pour la stabilité de la structure de la protéine et sa fonction.

2024-11-21

Travail exécuté par : MOSER Mathilda

Les weblogos du 20/11 sont difficile à interpréter car comportent toutes les positions de l'alignement ce qui ne donne pas d'informations globales sur les positions réellement conservées. Nous avons décidé d'alléger ces weblogos en ne gardant que les positions les plus conservés. Sur Jalview v2.11.4.1, nous avons sélectionné les positions qui possédait le plus d'occupancy. Le travail à été fait dans un premier temps à la main pour observer l'efficacité de cette méthode. Nous avons essayé d'avoir 20 à 30 aa par séquences pour garder les principales positions conservées des séquences.
De ces weblogos (PATH = "~/common/data/1_intermediate/weblogo/weblogo_crop/manually/filtered "), nous pouvons constater que le motif GH n'est pas toujours strictement conservé, de même pour le motif WD. Cependant nous pouvons observer un motifs récurent : GH-3*Hydophobe-Proline-Hydrophobe-G ou A-Acide-2*Apolaire-W-D. Nous observons aussi des différences entre chaque classe de repeats. Par exemple tous les repeats de «classes 5» semble ne pas posséder l'histidine qui est présente dans les autres.
Une possibilité serait que dans cette classe de repeats, l'histidine n'est pas nécessaire. Une autre possibilité vient du fait que le tri à été fait « à l'œil ». Nous avons pu passer à coté de quelque positions car ces dernières n'étaient pas bien conservés ou alignés.
En plus des logos obtenues par sélection manuelle, nous sommes en train de générer des logos obtenu grâce un algorithme de tri basé sur un seuil

d’occupancy crée par Guillaume.

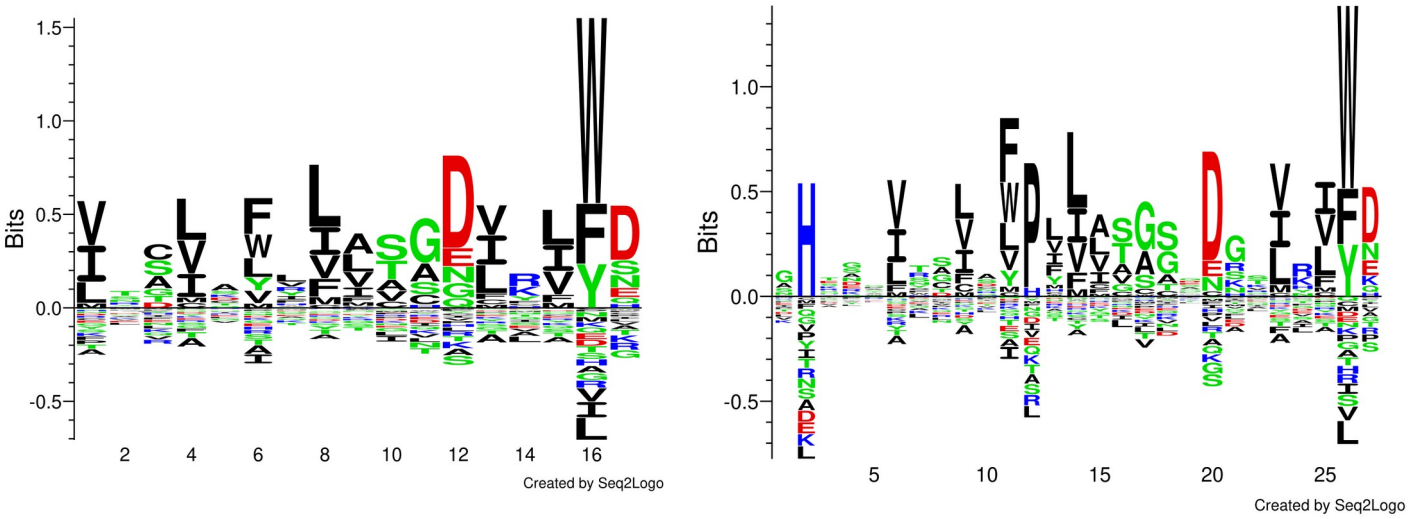


Figure 3 : Weblogos triés à la main. (Droite. classe 5, Gauche. classes 6)

Les weblogos ont étés générés grâce à l’outil en ligne Seq2Logo (https://services.healthtech.dtu.dk/services/Seq2Logo-2.0/) car la librairie python installable via pip3 ne nous fournissait pas de weblogos de qualité suffisante (voir script d’essai ~/common/scripts/gen_weblogo.py).

2024-11-18 à 2024-11-19

Travail exécuté par : WITTENMEYER Guillaume

Pour avoir une idée plus précise de ces weblogos, nous avons choisit de filtrer l’occupancy de chaque position dans ces alignements afin de pouvoir quantifier cette conservation dans l’alignement et éviter les biais humain.
Nous avons donc créer un script python permettant de créer ces nouveaux alignements filtrés en fonction de l’occupancy (~/common/scripts/occupancy.py).
Nous avons également choisi de filtrer ces différents alignements à plusieurs occupancy [0,6 ; 0,7 ; 0,8] (= fréquence d’apparition d’aa sans gaps à une position donnée) pour les comparer. Les alignements filtrés de sorties sont sauvegardées dans des sous-dossiers ‘/occupancy/’ pour chaque alignement intermédiaires contenus dans le dossier des alignements MAFFT ~/common/data/1_intermediate/MAFFT_alignment/.
Nous avons aussi codé dans ce script python une petite fonction permettant de rapidement visualiser des weblogos générés avec python en local (trop illisibles pour une étude poussée) (fonction se base sur un script test inutilisé dans le projet ~/common/scripts/gen_weblogo.py
Ces fichiers filtrés seront utilisés par Mathilda pour générer des logos plus visuels et compréhensibles en utilisant Seq2Logo (Les weblogos ont étés générés grâce à l’outil en ligne Seq2Logo (https://services.healthtech.dtu.dk/services/Seq2Logo-2.0/)).

Le script python a été exécuté dans l’environnement conda associé au projet ~/common/scripts/wd_repeat.yml

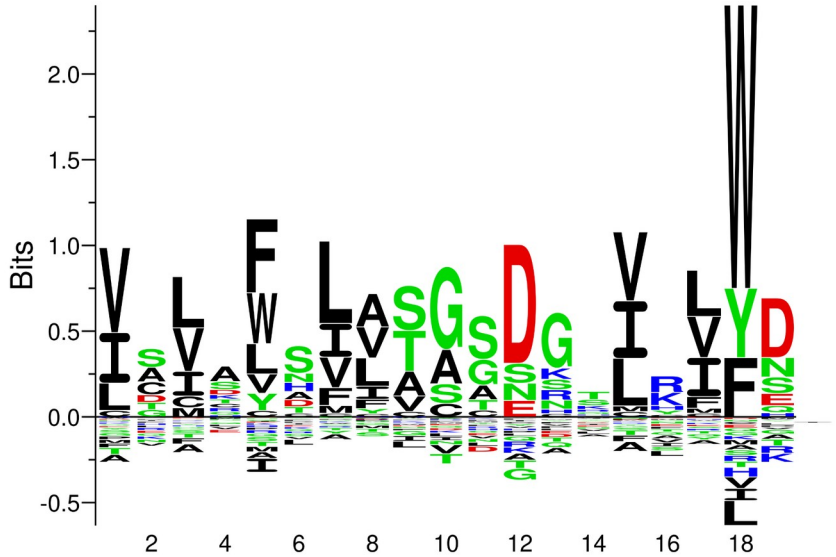


Figure 4A: Weblogo de toutes les classes pour une occupancy de 80%

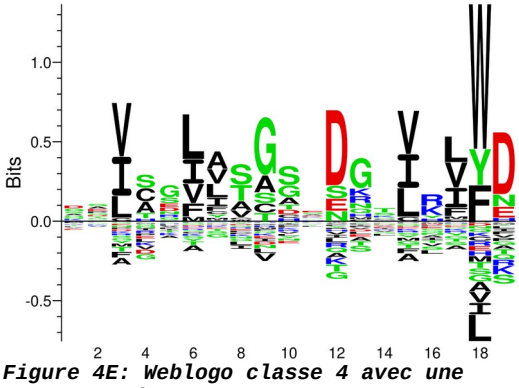


Figure 4E: Weblogo classe 4 avec une occupancy de 80%

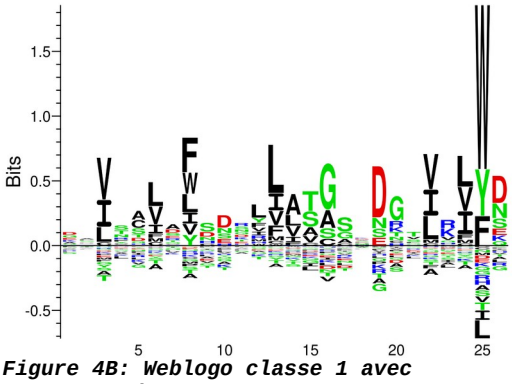


Figure 4B: Weblogo classe 1 avec occupancy de 80%

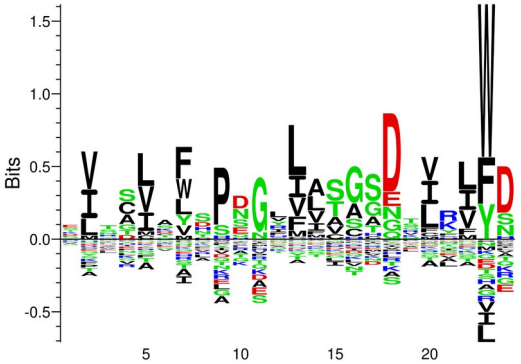


Figure 4F: Weblogo classe 5 avec une occupancy de 80%

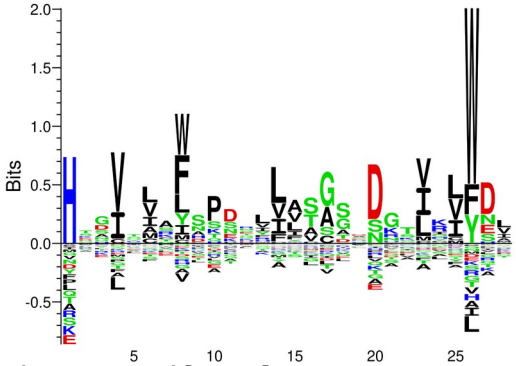


Figure 4C: Weblogo classe 2 avec occupancy de 80%

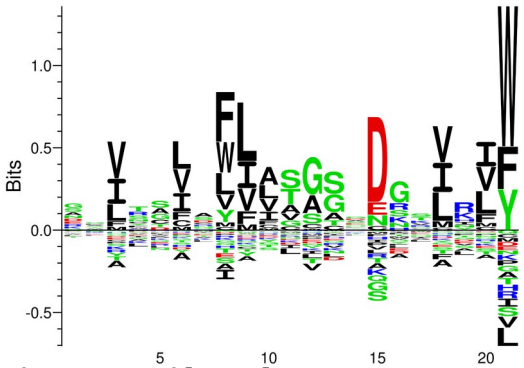


Figure 4F: Weblogo classe 6 avec une occupancy de 80%

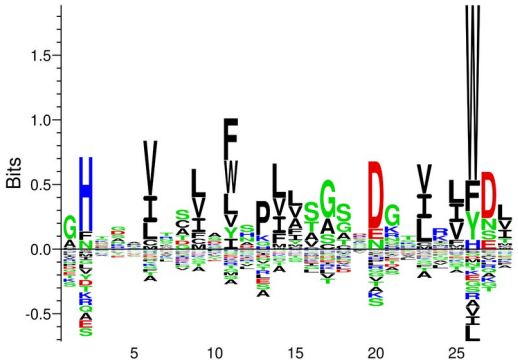


Figure 4D: Weblogo classe 3 avec occupancy de 80%

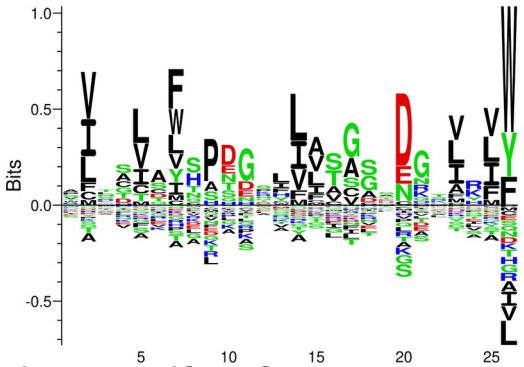


Figure 4G: Weblogo classe 7 avec une occupancy de 80%

On remarque donc à travers ces différents weblogos avec une occupancy de 80 % (choisie car c’est celle qui représente le mieux les positions bien conservées) que certains motifs sont conservés entre les différences classes de repeats et d’autres non. Le cas le plus flagrant est le cas de la conservation du motif GH qui ne semble conservé que par la classe 3 (figure 4D) et l’histidine uniquement pour la classe 2 (figure 4C). Cela pourrait donc expliquer que ce motif n’apparaît pas conservé avec une occupancy de 80 % sur le weblogo global (figure 4A).
On remarque également que seule la classe 7 (figure 4G) ne possède pas le motif WD dans son intégralité avec une occupancy de 80 %.
Néanmoins, on peut aussi souligné le fait que ces deux différences ne sont pas présentes pour une occupancy de 70 % (~/common/data/1_intermediate/weblogo/weblogo_crop/occupacy/filtered/ occupacy_07/no_extensions/). Ce qui pourrait alors nous faire penser que ce motif se trouve dans une région qui n’est pas forcément très conservée et qui pourrait donc mal s’aligner, on peut notamment penser aux boucles de liaisons entre chaque blades.
On peut également en extraire un motif récurent parmis toutes ces classes :

[V,I,L]-x-[L,V,I]-[F,W]-polaire-[L,I,V]-hydrophobe-[S,T]-G-S-D-G-[V,I,L]-[R,K]-[L,V,I]-W-D

2024-11-19 à 2024-11-20

Travail exécuté par : WITTENMEYER Guillaume

Pour la suite du projet, nous avons trouvé intéressant d’effectuer une analyse structurale de ces alignements. Cela nous permettrait de visualiser les positions intéressant et de potentiellement observer et étudier les différentes interactions effectuées par ces positions dans la structure inter et intra blade. Cela permettrait par exemple de faire des hypothèse sur la façon dont son conservées le positions ainsi que d’observer les différences de ces séquences.

Pour cela, nous avons, à partir du fichier filtré de la classe 7 (~/common/data/1_intermediate/MAFFT_alignment/WD_sequence_filtered/ MAFFT_WD_sequence_classfiltered_7.fasta) récupérer toute les structures pdb des protéines de notre alignement possédant 7 repeats au minimum et au maximum (soit 143 protéines sur les 252 disponibles dans l’étude (voir https://github.com/Hudego/PTU_Project_3.git)) grâce à un script python récupérant les structures sur AlphaFold v. ? ~/common/scripts/alphafold.py
Les structures récupérés sont stockées dans le dossier ~/common/data/0_raw/pdb/.
On remarque cependant que toutes les structures ne sont pas prédites par AlphaFold :

PDB ID
Q15751
Q6ZNJ1
Q6ZQQ6
Q99698

De plus, certaines structures possèdent plusieurs bêta-propellers :

PDB ID	Nbr of domaines prédits
O15040	3
O75717	2
P53621	2
P54198	1,5
Q9HBG6	2
Q9P2H3	1,5
Q96RY7	2

Cela peut représenter un problème si nous souhaitons effectuer une superposition des structures complète des domaines, introduisant un nombre considérable d'erreur dû à ce nombre variable de domaine.

Ou alors possèdent de grandes régions mal prédites par AlphaFold comme O15040 :

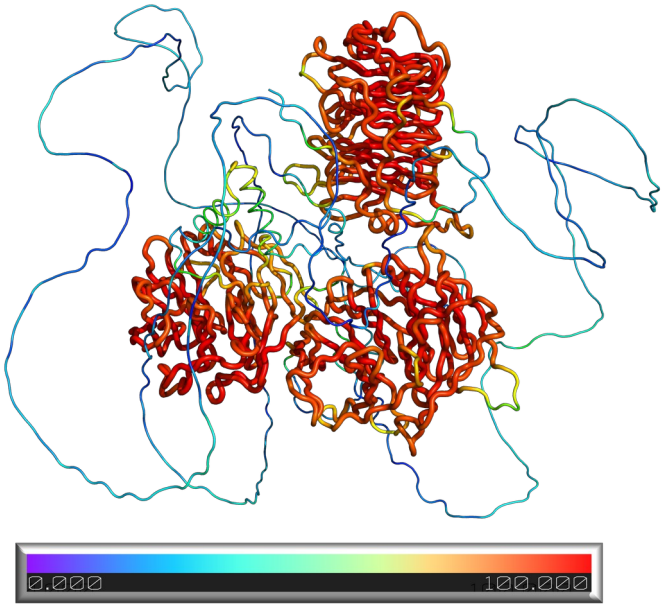


Figure 5: pLDDT visualisation avec PyMol

Il faut cependant faire attention car PyMol affiche les régions avec un taux de confiance de prédiction élevée (=pLDDT proche de 100) épaisses et rouges et les régions à faible taux de confiance de prédiction fines et bleues.

Ces structures représentent cependant les structures complète des domaines et nous nous intéressons aux différences entre chaque classe de repeats. Il faut donc décomposer ces structures en 1 structure par blade de chaque bêta-propeller.
Pour cela, nous avons codé un script python ~/common/scripts/cut_blades.py qui permet de récupérer les information de coordonnées et de longueur de chaque features de type repeat dans la séquence pour chaque protéine ~/common/data/0_raw/excel/WD_extracted_data.xlsx
Les fichier pdbs créés sont stockées dans des sous-dossiers pour chaques blades dans le dossier ~/common/data/1_intermediate/pdb_cut/

Le script python a été exécuté dans l'environnement conda associé au projet ~/common/scripts/wd_repeat.yml

2024-11-21 à 2024-11-23

Travail exécuté par : WITTENMEYER Guillaume

L'étape suivante est de pouvoir super-imposer ces structures de blades en fonction des classes dans PyMol.
Pour cela, nous avons codé un script python ~/common/scripts/superimpose.py qui permet de choisir la classe de repeat à super-imposer et qui super-impose ces structure dans une interface GUI de PyMol. Nous avons choisi de super-imposer avec la fonction super() de PyMol car les séquences des repeats étant souvent différentes, le fonction super() sera plus performante pour avoir des superpositions de structures cohérentes.

Cependant, on peut observer dans PyMol que toutes les super-impositions ne sont pas très efficaces comme la super-imposition du blade 6 de Q9Y4P3 avec le blade 6 de A2RU52 :

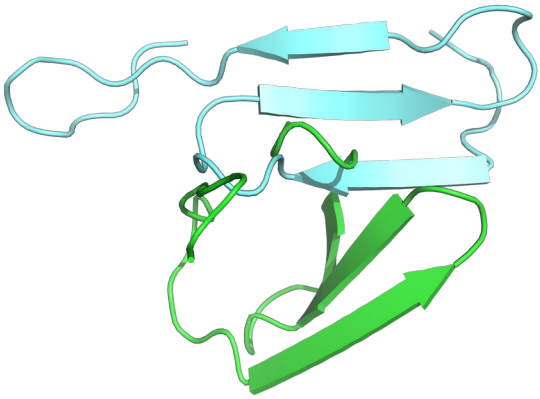


Figure 6: PyMol visualisation de super Q9Y4P3-A2RU52 classe 6

Nous avons donc eu l'idée de stocker grâce à ce même script les RMSD correspondant à chaque super-imposition ainsi que le nombre d'atomes pris en compte relatifs au calcul de ce RMSD pour pouvoir gérer des graphiques du nombre d'atomes pris en compte en fonction du log10(RMSD) dans des fichier txt pour chaque classe dans les sous dossiers de chaque cut per blade number dans dossier ~/common/data/1_intermediate/pdb_cut/
Pour cela, nous avons créer un script python ~/common/scripts/graphs.py qui génère ce graphiques et qui sont disponible dans ces même sous dossiers dans le dossier ~/common/data/1_intermediate/pdb_cut/

On obtient ce genre de graphique qui permet de voir la répartition des scores de RMSD en fonction du nombre d'atomes pris en compte dans la super-imposition et nous permets de potentiellement exclure les mauvaises super-imposition plus tard

dans notre projet :

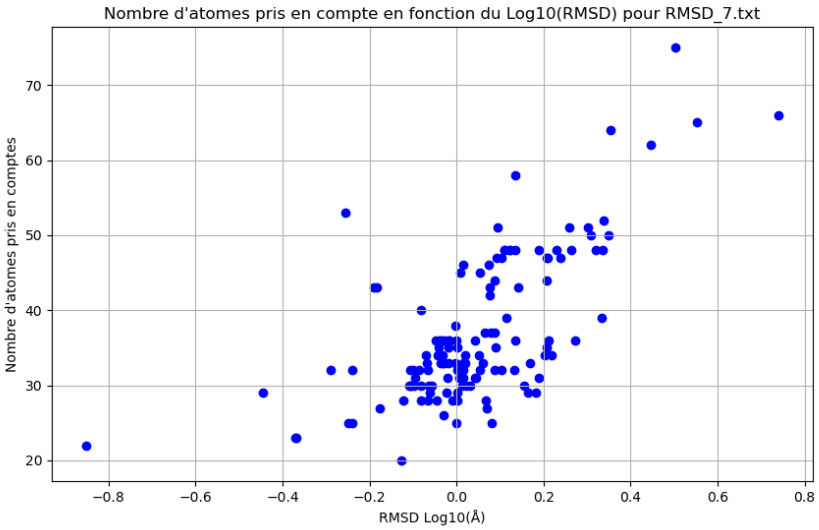


Figure 7B: Nbr d'atome pris dans la superimposition en fonction du log10(RMSD) pour la classe 7

Ces figures montrent bien que les classes numéro 4 et 7 a de très mauvais scores de superimposition dû fait du faible nombre d'atomes pris en compte dans cette superimposition.

2024-11-19 à 2024-11-20

Travail exécuté par : WITTENMEYER Guillaume

Suite aux analyses des alignements par blade effectuées par Mohsen, nous souhaitions pouvoir visualiser les structures dans d'autres logiciels de visualisation que PyMol. Pour cela, nous avons donc eu l'idée de sauvegarder les coordonnées des structures des blades super-imposées issus du script python ~/common/scripts/superimpose.py dans de nouveaux fichiers pdb pour pouvoir ouvrir et étudier ces structures dans d'autres logiciels de visualisation molécuaires tels que DiscoveryStudio.
Nous avons donc implémenté une nouvelle fonction python dans le script ~/common/scripts/superimpose.py pour effectuer cela et stocker ces fichiers dans des sous-dossiers pour chaque super-imposition de blades dans le dossier ~/common/data/1_intermediate/pdb_superstructure/

2024-11-28 à 2024-12-04

Travail exécuté par : WITTENMEYER Guillaume

Amélioration du script de superimposition (~/common/scripts/superimpose.py).

Pour chaque superposition de classe, nous allons aussi grâce à ce script déterminer la structure de référence pour la superimposition pour laquelle la moyenne du RMSD de chaque superimposition par rapport à cette structure est la plus petite. Pour cela, nous avons codé une petite fonction dans le script permettant de calculer ces averages (~common/scripts/graphs.py).

Class of WD repeats	Reference PDB ID	RMSD Avg	Atoms number pick during super() Avg
1	Q6ZMY6	1.5221855251023368	153.5182481751825
2	Q9UKT8	1.160593188374582	130.86861313868613
3	Q9NRL3	0.9477170253322073	157.02919708029196
4	Q9BRP4	2.0674683901950393	56.802919708029194
5	Q9Y4P8	1.2597052256266277	148.5
6	Q9Y4P8	1.421951580306758	141.7173913043478
7	P57081	1.2092456939446665	36.89051094890511

On peut donc voir qu'utiliser uniquement l'average des RMSD comme point de repère pour les superimposition ne suffit pas forcément car on peut observer que les classes 4 et 7 n'ont pas été superimposer sur un grand nombre d'atomes, tout en ayant un RMSD average plutôt élevé (classe 4) ou moyen (classe 7).

On obtiens ainsi les graphiques issus de ~/common/scripts/graphs.py pour chaque classe :

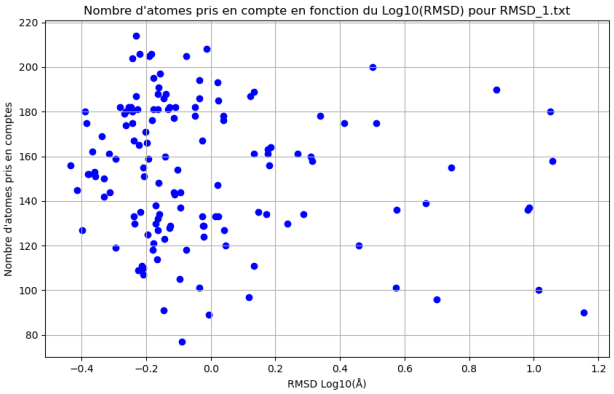


Figure 8A: Nbr at in super() en fonction du log10(RMSD) classe 1

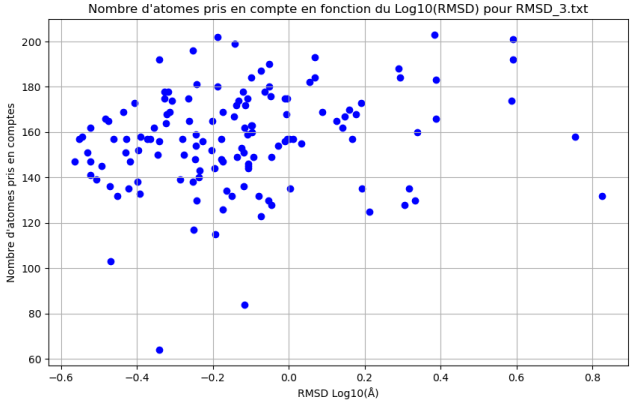


Figure 8C: Nbr at in super() en fonction du log10(RMSD) classe 3

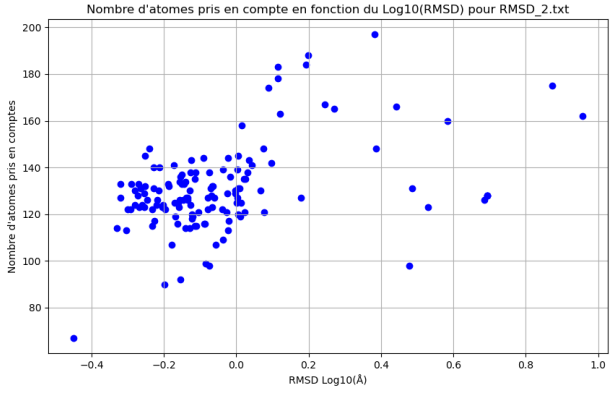


Figure 8B: Nbr at in super() en fonction du log10(RMSD) classe 2

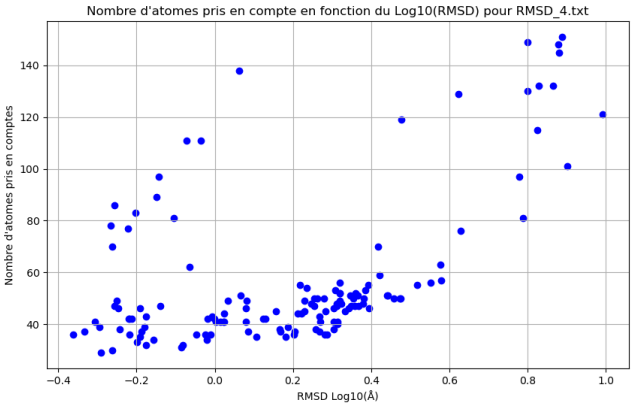


Figure 8D: Nbr at in super() en fonction du log10(RMSD) classe 4

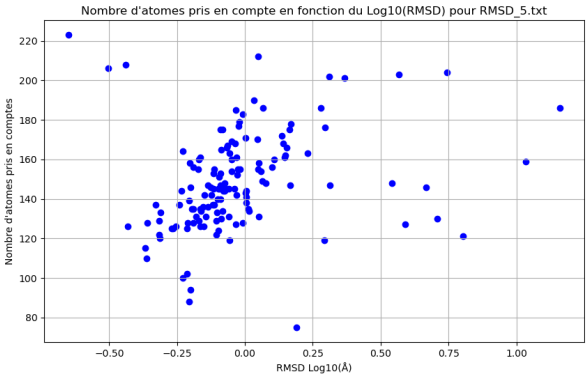


Figure 8E: Nbr at in super() en fonction du log10(RMSD) classe 5

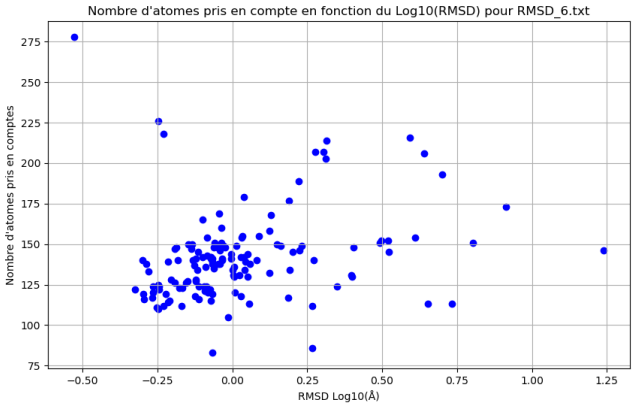


Figure 8F: Nbr at in super() en fonction du log10(RMSD) classe 6

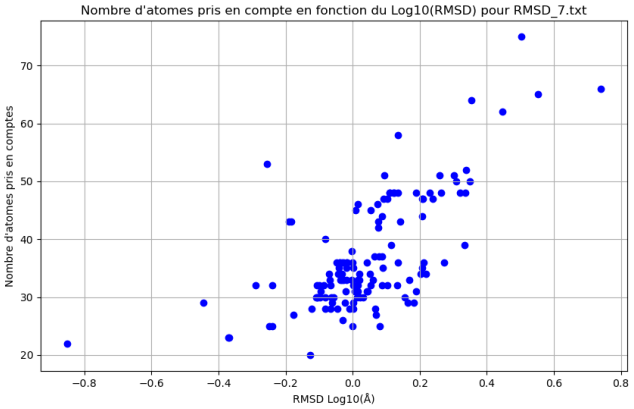


Figure 8H: Nbr at in super() en fonction du log10(RMSD) classe 7

On remarque grâce à ces graphiques que les superimposition de chaque classe de repeats possède des superimpositions très hétérogènes avec pour les classes 4 et 7 des répartitions très faibles, ce qui pourrait être un indicateur de mauvaises superimpositions. On pourrait aussi parler de la classe 1 qui semble avoir une répartition des superimpositions très hétéroclites, ce qui pourrait des traduire par des structures qui sont plus éloignées dans cette classe de repeats et donc qui s'aligneraient très différemment les unes des autres. Si l'on observe ces superpositions de classes de repeats (~/common/data/1_intermediate/pdb_superstructure/) que certains repeats ne se sont pas correctement superimposés.

Par soucis de temps dans le cadre de ce projet, nous n'avons pas pu effectuer d'analyses structurales plus approfondies. Nous avons néanmoins une idée de l'approche à effectuer :
En se basant sur les observations des excels produits par Mohsen, il faudrait créer un script ou modifier les scripts existants pour pouvoir garder en mémoire les positions résidus conserver dans les alignements et les transposer dans la structure des repeats pour pouvoir afficher ces résidus et étudier leurs différentes interactions et émettre des hypothèses sur les raisons de leur conservations.

2024-11- ?

Travail exécuté par : KESHAVARZ-NAJAFI Mohsen

Calcul de certaines statistiques basées sur les observation effectuées dans le tableur
~/common/data/1_intermediate/excels/Conservation_classfiltered_alignment.xlss
comme la variance intra et inter séquences de chaques classes de repeats.

For detailed guidance or troubleshooting, please refer to the full project documentation or contact the project lead.