



Projet de Recherche

**Regroupement et Visualisation de Protéines  
par les architectures de Domaines**

Par **Chadi Jaouadi** et **Paul Saïghi**

Encadrants :

**Juliana Silva Bernardes**

Maître de Conférences

Laboratoire Biologie Computationnelle et Quantitative (LCQB)

**Fabrice Kordon**

Professeur

Laboratoire Informatique Paris 6 (LIP6)

*UE 3I013 - Mai 2019*

# Preface

*Nous tenons à remercier vivement*

Mme Juliana Silva Bernardes pour son aide constante et son encadrement rigoureux

L'équipe du Laboratoire Biologie Computationnelle et Quantitative pour son accueil chaleureux réservé à chacun de nos passages dans ses locaux

M Fabrice Kordon pour son implication dans cette UE d'Initiation à la Recherche qu'on a trouvée passionnante :-)

# Table des matières

<b>1</b>	<b>Problématique</b>	<b>4</b>
<b>2</b>	<b>Outils, mise en oeuvre du projet</b>	<b>6</b>
2.1	Choix des outils . . . . .	6
2.2	Modélisation . . . . .	7
2.3	Aspects Algorithmiques . . . . .	8
2.3.1	Similarité entre deux protéines : Distance de Damerau-Levenshtein . . . . .	8
2.3.2	Similarité entre deux groupes de protéines : Lien Moyen . . . . .	9
2.3.3	Regroupement Hiérarchique . . . . .	10
2.4	Résultats vers tableur Excel : Librairie PhpSpreadSheet . . . . .	12
<b>3</b>	<b>Résultats</b>	<b>13</b>
3.1	Traitement de données textuelles saisies . . . . .	13
3.2	Visualisation interactive des protéines . . . . .	15
3.3	Regroupement par similarité d'architectures de domaines . . . . .	16
3.4	Sauvegarde des résultats en format Excel . . . . .	18
<b>4</b>	<b>Conclusion</b>	<b>19</b>
	<b>Bibliographie</b>	<b>20</b>

# Chapitre 1

## Problématique

La recherche et la classification des protéines homologues sont des enjeux biologiques importants. Une protéine est une molécule trouvée chez tous les êtres vivants.

Elles sont formées à partir de plusieurs acides aminés qui sont enfilés les uns après les autres dans un ordre précis. Cet ordre est donné par les gènes qui sont des fragments de la molécule d'ADN. Chaque protéine possède une fonction cellulaire. On peut citer quelques exemples de protéines ainsi que leurs fonctions principales :

- **L'hémoglobine** est une protéine présente dans les globules rouges du sang des vertébrés. Elle a pour fonction de transporter l'oxygène de l'appareil respiratoire vers le reste de l'organisme.
- **Les enzymes** sont des protéines ayant un rôle de catalyseur. Pratiquement toute biomolécule capable de catalyser des réactions chimiques dans une cellule est une enzyme.
- **Le glutamate** fait office de neurotransmetteur. Il est soupçonné de jouer un rôle dans les fonctions cérébrales d'apprentissage et de mémorisation.

Les protéines sont donc omniprésentes dans le vivant. Elles remplissent des rôles très divers et il reste à découvrir les fonctions d'un grand nombre d'entre elles. Une protéine peut être représentée par une chaîne de caractère de taille variable, où chaque lettre représente un acide aminé. Il y a 20 possibilités d'acide aminé.

Dans cette chaîne, ou séquence protéique, existent des parties appelées domaines qui sont capables d'adopter une structure autonome du reste de la molécule et jouent un rôle très important dans la fonction de la protéine. Les protéines peuvent ainsi être composées de l'assemblage de plusieurs domaines, dans ce cas elles sont appelées protéines multi-domaines.

Pour déterminer la fonction d'une nouvelle protéine, une étape primordiale est de comparer sa séquence d'acide aminé avec celle d'une protéine dont sa fonction est déjà connue (préalablement déposée dans une base de données).

Cette méthode suppose que les séquences présentant une similarité significative partagent une origine commune, c'est-à-dire qu'elles sont homologues.

Cependant, l'existence de protéines multi-domaines et de mécanismes évolutifs complexes pose des difficultés aux méthodes basées sur les comparaisons de séquences. L'alternative, qui constitue le sujet de ce projet, est une comparaison basée sur les domaines.

Dans ce cas, la séquence de la nouvelle protéine est comparée à une base de données de domaines, comme par exemple, la base PFAM ([Bateman, 2004](#)), et les positions de domaines sont identifiées. La Table 1.1 montre un exemple des domaines qui ont été identifiés pour un ensemble de protéines d'intérêt. Une protéine est décrite par une ligne et les domaines identifiés constituent son architecture.

Les deux premières colonnes informent sur la Protéine. Il s'agit respectivement de : l'identifiant de la protéine, et de sa taille. À partir de la 3ème colonne, les domaines qui constituent cette protéine sont décrits. Le domaine est représenté par un couple de 4 colonnes consécutives. Les colonnes représentent respectivement : identifiant du domaine, valeur de confiance de la prédiction du domaine, position où le domaine commence, position où le domaine fini dans la protéine. Les domaines montrés dans la table 1.1 ont été préalablement identifiés par un outil de prédiction de domaines développé par Mme Bernardes et ses collègues du LCQB ([J.S. Bernardes & Carbone, 2015](#)).

TABLE 1.1 – Identification de domaines

Coscinodiscus_wailesii	1300	PF00385	1.18e-14	39	76	PF00595	5.17e-09	305	342
Agaricus_bisporus	2500	PF00145	9.16e-120	272	599	PF01485	0.000158		
Amphiprora_paludosa	765	PF00145	8.62e-64	346	647				
Arabidopsis_thaliana	497	PF00145	5.78e-115	19	377				
Arabidopsis_thaliana	870	PF00145	3.79e-10	306	491	PF00145	3.88e-43	502	622
Aspergillus_arachidicola	1234	PF01426	3.73e-30	138	250	PF00145	3.07e-80	321	591
Aspergillus_niger	611	PF01426	2.83e-31	132	240	PF00145	5.62e-87	306	513

Les représentations textuelles de protéines, illustrées dans la table 1.1, ne sont pas adaptées aux travaux de comparaisons et d'analyses des biologistes. Pour permettre au biologiste de visualiser de manière graphique et interactive les protéines et leurs architecture de domaines, nous avons développé dans un premier temps un serveur web. Dans un second temps, nous avons implémenté un algorithme de regroupement pour regrouper les protéines ayant des architectures de domaines similaires afin de rendre leur comparaison et leur analyse d'autant plus simple.

Nous avons également intégré plusieurs fonctionnalités pour répondre aux besoins d'utilisations pratiques des biologistes : sauvegarde des résultats sous format Excel et plus d'interactivité, comme la pagination de protéines sans devoir recharger la page, masquer et démasquer des groupes de protéines, etc.

# Chapitre 2

## Outils, mise en oeuvre du projet

### 2.1 Choix des outils

Nous nous sommes toujours appuyés sur les outils tels que Github pour l'organisation, SVG pour la visualisation graphique et le paradigme orienté objet en PHP pour structurer et développer l'ensemble des fonctionnalités. Dans cette seconde partie du semestre, nous avons apporté des améliorations en utilisant de nouveaux outils : JavaScript et HTML5 / CSS.

#### *PHP et librairies*

Comme la plus part des algorithmes qui nous intéressaient n'était pas disponibles dans des bibliothèques PHP, nous avons fait le choix d'implémenter certains algorithmes à partir de pseudo codes. Ce choix nous épargne de recourir à des techniques plus lourdes tel que des configurations serveur pour interagir entre du PHP et du code Python. En revanche, pour sauvegarder les résultats sous format Excel, la librairie PhpSpreadSheet <sup>1</sup> a répondu à nos attentes.

#### *JavaScript et interactivité*

Pour améliorer interactivité de la visualisation graphique des protéines, nous nous sommes basés sur le JavaScript pour pouvoir masquer et démasquer grâce à un bouton l'ensemble des protéines d'un groupe. Nous ajoutons aussi à cela une façon plus adaptée de passer d'une page à une autre : désormais nous n'avons plus à recharger la page et donc à recalculer les données, pour obtenir un nouveau résultat. Pour ce faire, nous manipulons avec JavaScript des noeuds du Document Object Model (DOM) (Jeremy Keith, 2011) .

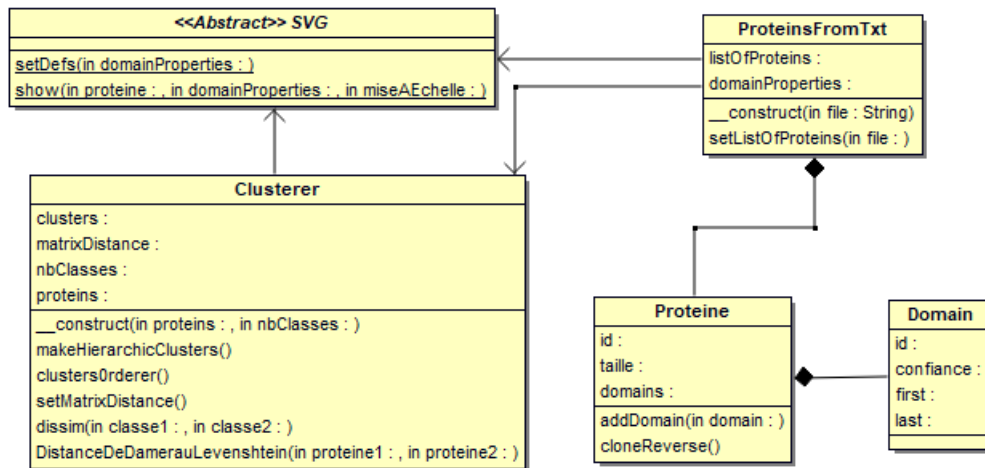
---

1. PhpSpreadsheet's documentation - <https://phpspreadsheet.readthedocs.io/en/latest/>

## *Bootstrap, HTLM5 et CSS : Design et mise en page*

Nous utilisons un nouvel outil conjointement au HTML5 et CSS, Bootstrap ([Philibert, 2015](#)). C'est une collection d'outils, mêlant le CSS et le JavaScript, utiles à la création du design de sites et d'applications web.

## 2.2 Modélisation



**Fig. 2.1** – *Diagramme de Classes UML*

Au regard de ces complétions et ces améliorations de notre outil, une nouvelle modélisation a été conçue comme représenté au diagramme UML de la Figure 2.1.

Pour rappel, le biologiste rentre le fichier de données textuelles représentant les protéines et leurs architecture de domaines au moyen d'un formulaire. Nous créons alors un objet **ProteinsFromTxt** qui va parser ce fichier texte et transformer chaque ligne en **Proteine** munie de sa liste d'objets **Domain**, c'est-à-dire de son attribut **domains**.

Le parseur **ProteinsFromTxt** communique avec deux classes : **SVG** et **Clusterer**. Au fur et à mesure du parsing, on définit à chaque **Domain** ses propriétés graphiques spécifiques, sur la base des algorithmes détaillées dans notre rapport de mi-parcours. Ces propriétés sont stockées dans l'attribut **domainProperties**. **ProteinsFromTxt** communique ces propriétés au singleton **SVG**. Ce dernier injectera, avec la méthode **setDefs**, les propriétés graphiques des domaines dans la balise "<def>" du noeud **SVG** de la page.

À l'issue du parsing, `ProteinsFromTxt` aura pour attribut `listOfProteins`, la liste contenant les protéines. Cette dernière sera soumise à l'objet `Clusterer` qui lui déroulera les algorithmes nécessaires de regroupement hiérarchique ([Rui Xu, 2008](#)).

À l'aide de la liste des protéines transmise par le parseur `ProteinsFromTxt` et d'un nombre défini de clusters à générer, `Clusterer` ordonnancera les protéines au sein de groupes appelés clusters. Le nombre de clusters est défini par l'utilisateur. Ce dernier est aidé dans ce choix par l'affichage d'un arbre appelé dendrogram.

Nous détaillerons par la suite les algorithmes utilisés dans ce processus de regroupement de protéines en fonction de leur similarités.

Enfin, à l'aide des clusters de `Clusterer`, on affichera graphiquement chaque protéine via la méthode `show` du singleton `SVG`, compte tenu automatiquement des propriétés spécifiques à chaque domaines ayant été préalablement initialisées.

## 2.3 Aspects Algorithmiques

### 2.3.1 Similarité entre deux protéines : Distance de Damerau-Levenshtein

Avant d'expliquer l'algorithme plus en détails il faut expliciter que chaque protéine est représentée par une chaîne de domaine. Par exemple dans la table [1.1](#), la protéine `Aspergillis_arachidicola` est représentée par la chaîne "AB", "A" correspond au premier domaine et "B" au second, la protéine `Arabidopsis_thaliana` par la chaîne "AA".

Pour mesurer la similarité entre deux protéines, nous sommes passés par une phase de recherche de l'algorithme de mesure de distance le plus adapté.

Premièrement nous avons étudié la distance de **Jaccard**. Celle-ci mesure l'intersection entre deux ensembles. Ainsi "ABC" et "BCA" ont une distance de 0 au sens de Jaccard car ils représentent les même ensemble de lettres. Jaccard n'est donc pas adapté. Il est nécessaire de pouvoir mesurer un changement d'ordre de domaines dans une protéine.

La seconde distance étudiée fût celle de **Levenshtein**. Elle prend compte la substitution d'un caractère de "A" par un caractère différent de "B", ajout dans "A" d'un caractère de "B", suppression d'un caractère. Ainsi la chaîne "ABC" est a une distance de 1 de la chaîne "ACB" et de la chaîne "AB". Or nous cherchons à ce que "ABC" soit considéré comme plus proche de "ACB" que de "AB", car la suppression et l'ajout de lettres/domaines change grandement la fonction de la protéine.



Au terme de ces comparaisons, nous avons fait le choix de la distance de **Damerau-Levenshtein**, pseudo-code disponible dans la page Wikipédia en référence<sup>2</sup>.

Celle-ci prend en compte la transposition de deux lettres. Cet algorithme calcule le nombre minimum d'opérations nécessaires pour transformer une chaîne de caractères en une autre. Une opération est définie comme l'insertion, la suppression ou la substitution d'un simple caractère, ou comme une transposition de deux caractères adjacents.

Il est à noter que changer les caractères de place "coûte" moins cher que d'en supprimer ou d'en rajouter, cela pèse moins lourd pour le problème d'optimisation.

Par exemple, la chaîne "ABC" est plus proche de la chaîne "ACB" que de la chaîne "AB" ou que de la chaîne "AC" au sens de Damerau-Levenshtein.

Il nous semble que cette mesure de distance soit la plus adaptée et intuitive pour comparer deux protéines. Par ailleurs, la distance de Damerau-Levenshtein est utilisée dans plusieurs études dans le domaine de la bio-informatique, comme par exemple dans les variations de séquences d'ADN (Alexander Bolshoy, 2010).

### 2.3.2 Similarité entre deux groupes de protéines : Lien Moyen

L'algorithme de Damerau-Levenshtein implémenté précédemment mesure la similarité entre deux protéines.

Si nous avons à comparer un groupe A et un groupe B, et que chacune ne contienne qu'une protéine, il est évident que la mesure soit le résultat de la distance de Damerau-Levenshtein entre la protéine du groupe A et la protéine du groupe B.

Maintenant, si un des deux groupes contient plusieurs protéines, comment calculer la similarité entre les deux groupes ?

Nous avons fait le choix du critère de lien moyen<sup>3</sup> : calculer la moyenne des similarités entre les protéines des deux groupes. Cette approche sera utile pour calculer la similarité entre deux groupes de protéines dans le cadre du regroupement hiérarchique, détaillé dans la partie suivante ainsi que dans le trie inter-groupe effectué pour l'affichage final des groupes de protéine.

Nous avons implémenté cet algorithme de distance inter-cluster dans la méthode `dissim` de l'objet `Clusterer`.

---

2. Damerau-Levenshtein distance, Optimal string alignment distance [https://en.wikipedia.org/wiki/Damerau-Levenshtein\\_distance#Optimal\\_string\\_alignment\\_distance](https://en.wikipedia.org/wiki/Damerau-Levenshtein_distance#Optimal_string_alignment_distance)

3. Lien Moyen - Mesure de dissimilarité inter-groupe [https://fr.wikipedia.org/wiki/Regroupement\\_hiérarchique#Mesure\\_de\\_dissimilarité\\_inter-classe](https://fr.wikipedia.org/wiki/Regroupement_hiérarchique#Mesure_de_dissimilarité_inter-classe)

### 2.3.3 Regroupement Hiérarchique

Avant d’afficher les protéines à l’aide du SVG, on les regroupe en clusters en fonction de leur similarités. Chaque cluster contiendra des protéines relativement similaires ce qui permettra au biologiste de trouver rapidement les protéines ayant une architecture proche. L’algorithme de clustering que nous avons utilisé est l’algorithme hiérarchique<sup>4</sup> qui établit un arbre, appelé Dendrogram, comme montré dans la Figure 2.2.

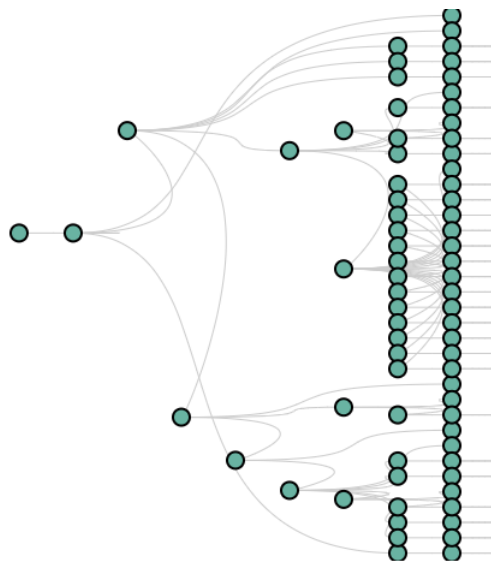
L’algorithme prend en entrée la liste totale de protéines et le nombre de clusters à obtenir, nbClusters.

Dans un premier temps, il crée un cluster par protéine.

Dans un second temps il applique une suite d’instructions en boucle, jusqu’à obtenir le nombre de clusters voulu, nbClusters. Un tour de boucle consiste à :

- Créer une matrice de comparaison des différents clusters.
- Détecter les deux clusters qui se ressemblent le plus, via cette matrice.
- Fusionner les deux clusters les plus similaires en un seul cluster

À la fin nous obtenons un ensemble de nbClusters groupes de protéines qui sont similaires entre elles.



**Fig. 2.2** – Exemple de Dendrogram

L’arbre de la Figure 2.2 représente les différents clusters qu’il est possible d’extraire. L’utilisateur s’aide de cet arbre pour choisir un nombre opportun de clusters à regrouper.

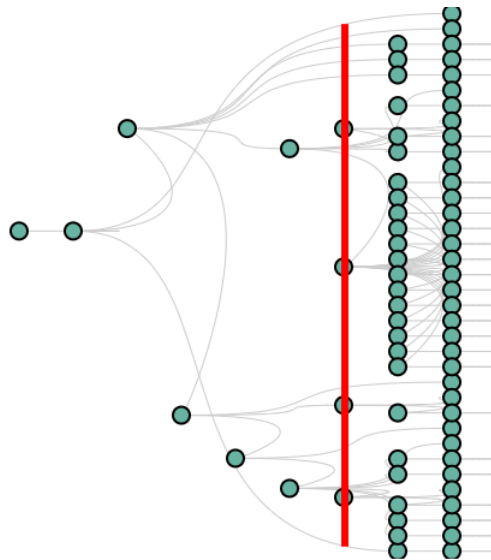
4. Regroupement Hiérarchique - Implémentation en pseudo-code.

[https://fr.wikipedia.org/wiki/Regroupement\\_hiérarchique#Implémentation\\_en\\_pseudo-code](https://fr.wikipedia.org/wiki/Regroupement_hiérarchique#Implémentation_en_pseudo-code)

Dans cet exemple, il serait pertinent de vouloir 4 clusters car au delà de 4 clusters (plus à droite dans l'arbre) les clusters sont trop proches et la classification qu'ils représentent ne semble pas avoir de sens.

Nous avons ré-implémenté en PHP le pseudo-code présent sur Wikipedia.<sup>5</sup> Nous évitons alors de recourir à des bibliothèques et configurations lourdes sous python ou PHP. Cet algorithme prend en entrée la liste de protéines et le nombre de cluster à produire. On utilise la fonction `makeHierarchicClusters` de l'objet `Clusterer` pour générer une liste de clusters (qui sont eux même des listes de protéines).

Une fois les clusters fait, nous les mettons dans une liste, cette liste est ordonnancée par la méthode `clusterOrderer`. Les clusters similaires sont mis les uns à côté des autres en fonction de leur ressemblances. Cette liste est stockée dans l'attribut `clusters` de `Clusterer`.



**Fig. 2.3** – *Exemple de Dendrogram coupé*

---

5. Regroupement Hiérarchique - Implémentation en Pseudo-Code [https://fr.wikipedia.org/wiki/Regroupement\\_hiérarchique#Implémentation\\_en\\_pseudo-code](https://fr.wikipedia.org/wiki/Regroupement_hiérarchique#Implémentation_en_pseudo-code)

## 2.4 Résultats vers tableur Excel : Librairie PhpSpreadSheet

Une fois que les résultats de regroupement sont obtenus, le chercheur a besoin de les conserver. Extraire les informations vers un tableur au format Excel est utile pour le biologiste afin de travailler sur ces résultats à tout moment sur sa machine.

Pour cela, utilisons la librairie PhpSpreadSheet pour générer le fichier sous format Excel. Cette bibliothèque offre un ensemble de classes permettant de lire et écrire dans différents formats de fichier de feuille de calcul.

Nous avons aussi pris soin de l'interactivité de ce fichier pour permettre à l'utilisateur de garder une vision globale sur la similarité des protéines.

Les cellules de domaines ont chacune des propriétés graphiques spécifiques. Ce sont des combinaisons de Formes de bordures + Couleur Bordure, de fond + Couleur de Fond et du format d'écriture : gras, italique, ou souligné...

Nous reprenons dans ce contexte les techniques algorithmiques habituelles de sélection de couleurs utilisées précédemment.

# Chapitre 3

## Résultats

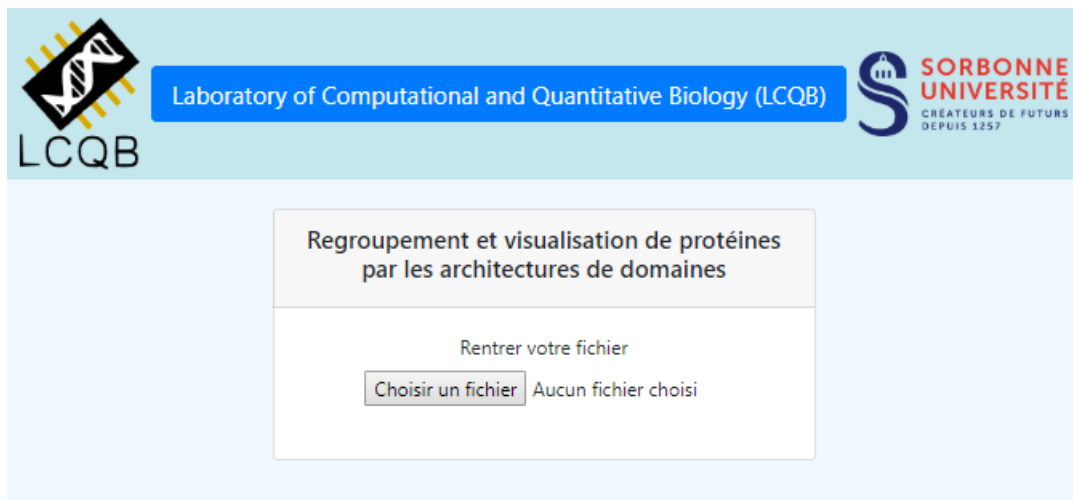
### 3.1 Traitement de données textuelles saisies

La récupération et le traitement des données sont faites suite à l'envoi du fichier textuel au serveur web.

Nous retrouvons dans la Figure 3.1 l'interface utilisateur comprenant le formulaire de saisie du fichier. Suite au chargement du fichier, un arbre Dendrogram, comme montré dans la Figure 3.2, est générée instantanément dans la page pour aider l'utilisateur à se décider sur le nombre de clusters qu'il veut obtenir.

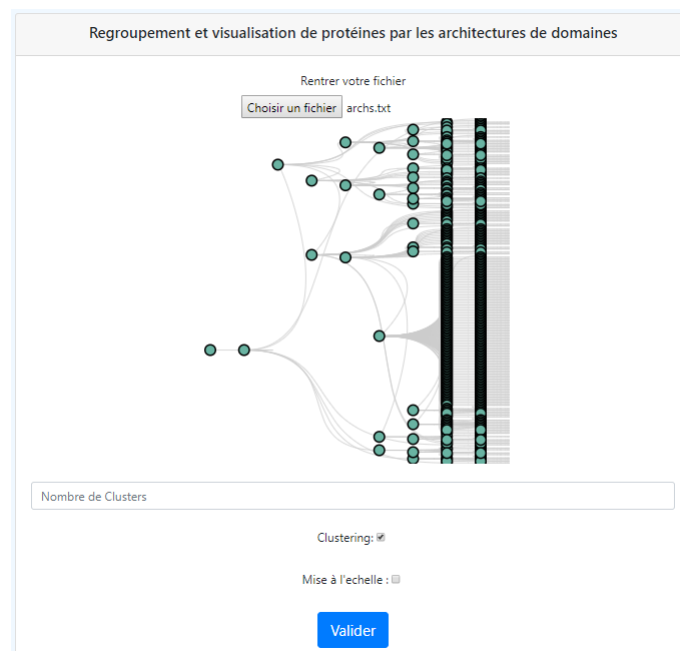
Pour le mode Clustering, l'utilisateur saisit le nombre de clusters qu'il souhaite. L'enjeu de l'utilisateur est de couper le Dendrogram à l'endroit qui lui paraît le plus opportun, pour déduire un nombre convenable de clusters à regrouper. Il doit s'aider du Dendrogram.

Le mode sans Clustering, admet que l'utilisateur saisisse 1 comme nombre de clusters souhaité. On retournera alors une seule liste de protéines triées en fonction de leurs ressemblances. En décochant la checkbox Clustering, le champs de saisie du nombre de clusters disparaît.



The screenshot shows the top header with the LCQB logo (a stylized DNA helix) and the text "Laboratory of Computational and Quantitative Biology (LCQB)". To the right is the Sorbonne University logo with the text "SORBONNE UNIVERSITÉ" and "CRÉATEURS DE FUTURS DEPUIS 1257". Below the header, a central box contains the title "Regroupement et visualisation de protéines par les architectures de domaines". Underneath the title, it says "Rentrer votre fichier". There is a button labeled "Choisir un fichier" and a status indicator that says "Aucun fichier choisi".

**Fig. 3.1** – *Interface Utilisateur - Formulaire*

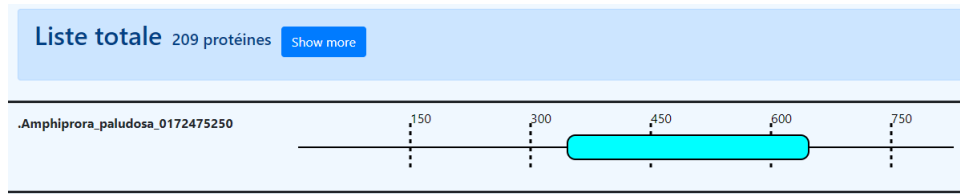


**Fig. 3.2** – *Dendrogramme généré après la saisie du fichier de données, contenant 209 protéines*

## 3.2 Visualisation interactive des protéines

Nous nous positionnons dans le cas où l'utilisateur souhaite visualiser toutes les protéines dans une seule liste. Entre autres, il ne souhaite obtenir qu'un seul cluster.

Nous obtenons alors une liste comportant toutes les protéines du fichier saisie. La liste est affichée avec la protéine située en tête de liste, comme l'illustre la Figure 3.3.

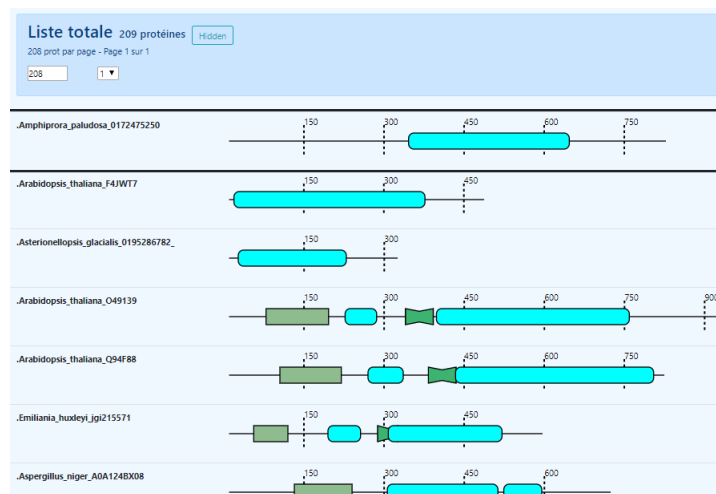


**Fig. 3.3** – Résultat pour un fichier contenant des données de 209 protéines, nous avons choisi d'obtenir une seule liste

Pour afficher l'ensemble des autres protéines de la liste, il suffit de cliquer sur le bouton Show More.

On peut remasquer le contenu avec le bouton Hidden. La Figure 3.4 présente la visualisation des protéines au sein de cette liste.

Les protéines sont triées par leur degré de ressemblances au sens de Damerau-Levenshtein, par rapport à la protéine en tête de liste.



**Fig. 3.4** – Bouton Show More - Affichage de l'ensemble des protéines de la liste

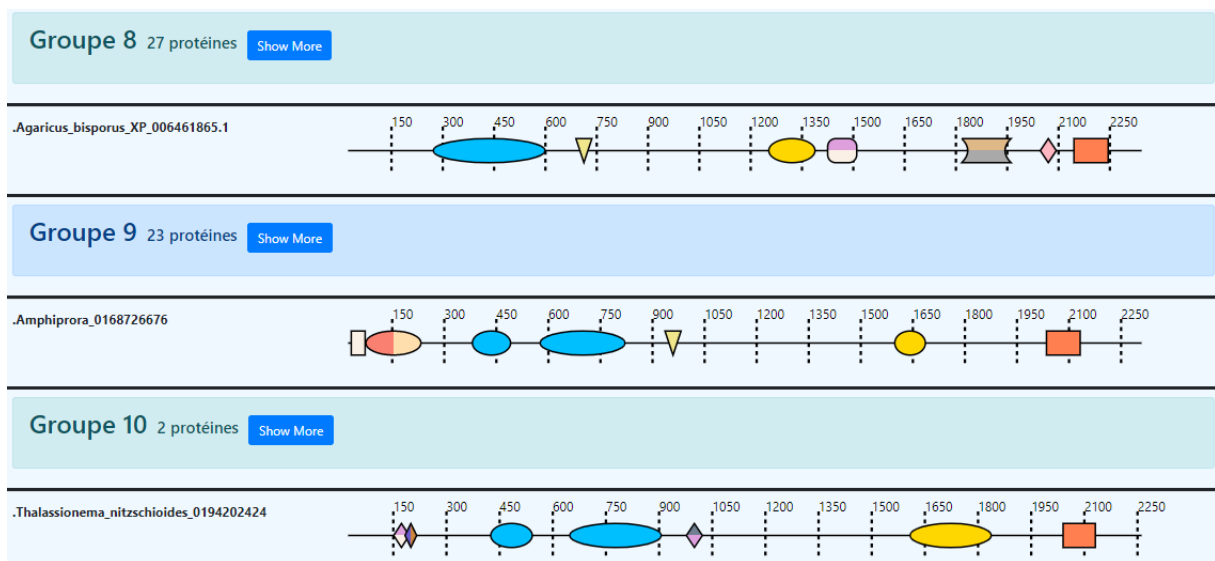
### 3.3 Regroupement par similarité d'architectures de domaines

La Figure 3.5 montre l'extrait du résultat sur un fichier contenant 209 protéines, nous avons fait le choix de produire 16 clusters. En cliquant sur Show more, on visualise l'ensemble de protéines du cluster triées en fonction de leur ressemblances à la protéine en tête de la liste du cluster.

Nous avons aussi mis en place un outil permettant de rendre la comparaison d'autant plus simple : la pagination, voir un exemple dans la Figure 3.6.

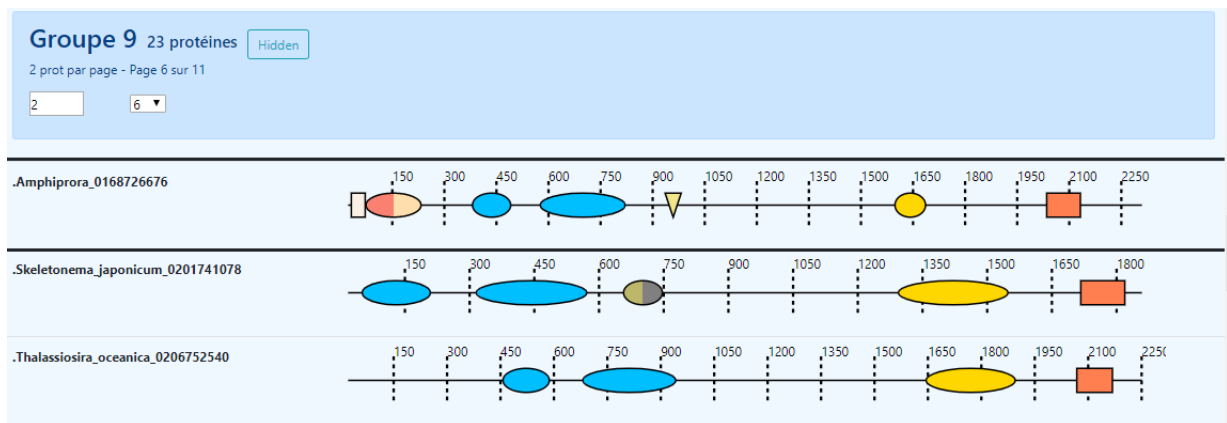
Elle permet de passer d'une page à une autre de protéines triées en fonction de la protéine référente du groupe. Nous gardons la protéine référente dans son même emplacement, en tête de liste. Grâce à JavaScript, il n'est plus nécessaire de ré-exécuter tous les programmes pour passer d'une page à une autre.

On peut aussi limiter le nombre de protéines par page à sa convenance, puis naviguer d'une page à une autre. Chaque changement à ce terme conduit à une modification du DOM par JavaScript. Aucun rechargement de la page n'est nécessaire. Nous gagnons alors beaucoup d'interactivité pour le biologiste, ce qui l'aidera sans doute à être plus productif dans ses recherches.



**Fig. 3.5** – Extrait du résultat sur un fichier contenant 209 protéines, nombre de clusters choisi : 16





**Fig. 3.6** – Exemple d'utilisation de la pagination : se limiter à comparer la protéine de référence du groupe 9 à 2 protéines, nous naviguons interactivement dans la page numéro 6

### 3.4 Sauvegarde des résultats en format Excel

Pour extraire les résultats sous format Excel, il suffit de cliquer sur le bouton vert en haut de la page, voir Figure 3.7. Il est aussi possible de pouvoir nommer le fichier excel qu'on va télécharger. Par défaut, il est initialisé au nom du fichier saisie mais il lui sera attribuée l'extension excel nécessaire. Un extrait est aperçu dans la Figure 3.8.

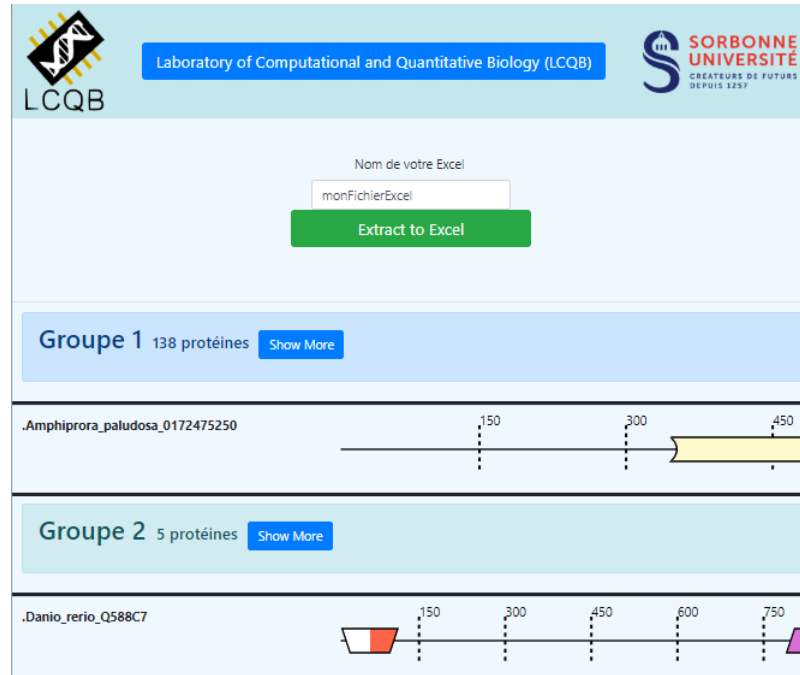


Fig. 3.7 – Bouton permettant d'extraire les données dans un fichier tableur au format Excel

Groupe 12 (4 protéines)													
Danio_rerio_Q588C7	1409	PF00307	7,91E-35	16	113	PF00855	1,79E-17	790	848	PF00628	8,47E-07	1066	1116
Homo_sapiens_Q9UBC3_	851	PF00855	2,93E-31	222	295	PF00628	1,93E-06	478	530	PF00145	1,67E-19	576	760
Homo_sapiens_Q9Y6K1_	973	PF10310	1,68E-25	1	172	PF00855	4,28E-29	290	364	PF00628	9,1E-06	536	586
Mus_musculus_Q88508	988	PF10310	1,7E-22	1	168	PF00855	3,73E-29	286	360	PF00628	8,74E-06	532	582
Groupe 13 (10 protéines)													
Ascobolus_immersus_Q42731	1426	PF12047	5,4E-42	93	248	PF01426	6,6E-22	432	575	PF01426	1,11E-09	638	744
Colletotrichum_gloeosporioides_L2GAQ2	1130	PF12047	8,28E-34	173	266	PF01426	1,68E-33	450	539	PF01426	1,26E-07	564	637
Cordyceps_militaris_G3JGU4	1147	PF12047	6,81E-44	158	267	PF01426	6,78E-35	436	527	PF01426	1,07E-06	554	656
Paracoccidioides_lutzi_C1H2T7	1274	PF12047	3,53E-47	189	296	PF01426	2,38E-31	458	547	PF01426	1,38E-09	569	656
Pseudogymnoascus_verrucosus_A0A1B8GCG9	1305	PF12047	1,2E-36	164	258	PF01426	3,88E-29	455	544	PF01426	1,99E-06	582	680
Purpureocillium_lilacinum_A0A179HTB7	1150	PF12047	1,34E-40	168	276	PF01426	5,13E-35	448	537	PF01426	1,14E-05	608	665
Rhizophagus_irregularis_A0A1C9IHL2	1373	PF12047	1,46E-30	208	352	PF01426	9,9E-30	514	626	PF01426	4,53E-20	683	825
Grosmannia_clavigera_F0XN91	1160	PF12047	1,06E-39	90	197	PF01426	1,55E-33	359	456	PF00145	4,9E-108	656	1056
Pochonia_chlamydosporia_A0A179FKN0	990	PF12047	4,21E-38	16	112	PF01426	4,84E-36	290	383	PF00145	1,5E-107	545	940
Neurospora_crassa_Q3Y3Z1	1296	PF12047	8,94E-45	218	324	PF01426	1,82E-48	575	679	PF00567	8,37E-05	666	721
Groupe 14 (6 protéines)													
Bos_taurus_Q24K09	1626	PF06464	7,61E-45	17	106	PF12877	1,51E-18	124	328	PF12047	5,4E-69	397	531
Homo_sapiens_P26358	1614	PF06464	8,47E-45	16	105	PF12877	1,7E-24	125	354	PF12047	4,86E-69	399	533
Ratus_norvegicus_Q9Z330	1619	PF06464	7,54E-45	16	106	PF12877	9,7E-10	93	344	PF12047	5,53E-65	405	539

Fig. 3.8 – Extrait du Fichier Excel généré à partir des résultats

# Chapitre 4

## Conclusion

Ce projet a été une opportunité enrichissante pour nous immerger dans le monde de la recherche bio-informatique au sein du Laboratoire de Biologie Computationnelle et Quantitative (LCBQ).

L'informatique est sans aucun doute d'utilité imminente pour la résolution des problèmes algorithmiques complexes des sciences de la vie.

Nous souhaitons que cette application soit d'agréable usage aux bio-informaticiens, aux biologistes, et particulièrement à l'ensemble de l'équipe de recherche du LCQB.

Nous souhaitons également que ces outils permettront, dans un avenir proche, de mener d'importantes découvertes biologiques et médicales.

Nous demeurons disponibles pour toutes questions et aides ultérieures pour le développement de ces outils.

# Bibliographie

- ALEXANDER BOLSHOY, ZEEV VOLKOVICH, K. 2010 Genome Clustering : From Linguistic Models to Classification of Genetic Texts. *Springer-Verlag Berlin Heidelberg* .
- BATEMAN, A., C. L. D. R. F. R. D. H. G.-J. S. S. D. J. 2004 The Pfam protein families database. *Nucleic acids research*, 32(suppl<sub>1</sub>), D138 – D141..
- JEREMY KEITH, J. S. 2011 DOM Scripting : Web Design with JavaScript and the Document Object Model. *APRESS* .
- J.S. BERNARDES, F.R.J. VIEIRA, G. Z. & CARBONE., A. 2015 A multi-objective approach accurately resolves protein domain architectures. *Bioinformatics* .
- PHILIBERT, B. 2015 Bootstrap 3 : Le framework 100% Web Design. *EYROLLES* .
- RUI XU, D. W. 2008 Clustering. *IEEE Computational Intelligence Society* .