LEBIB Inès – M2 BI Projet court Clustal

13/09/2023

Lien du Github : https://github.com/InLe074/LEBIB_Ines_Projet_Clustal_M2BI

Rapport : Alignement multiple heuristique par la méthode Clustal

L'objectif de ce projet est de réaliser un script Python qui reprend une méthode décrite dans un article écrit par Desmond G. Higgins et Paul M. Sharp en Avril 1989. Les auteurs présentent une méthodologie d'alignement de séquences multiple heuristique avec la méthode Clustal. Durant ce projet, nous allons nous intéresser exclusivement à l'alignement de séquences protéiques.

I. Introduction

La nécessité d'aligner automatiquement plusieurs séquences entre elles a fortement augmenter au cours de ces dernières décennies. En effet, il est possible d'extraire de précieuses et diverses informations grâce à un alignement multiples de séquences. Il est possible de comparer l'évolution et la structure d'un ensemble de séquences, de rechercher des sites ou des motifs conservées, de trouver des signatures de famille protéiques, jusqu'à 'prédire' des informations fonctionnelles et structurales (hélices, brins, boucles pour les structures secondaires et modélisation par homologie pour les structures tertiaires).

De nombreuses stratégies ont été mis en place afin de faire un alignement multiple. Nous allons nous nous intéresser à la méthode de programmation dynamique de Needleman & Wunsch (1970) et la méthode UPGMA pour construire un arbre enraciné (Sneath et Sokal, 1973).

II. Algorithmes

Les séquences protéiques que nous allons étudier sont téléchargées sur la base de données Uniprot au format FASTA. Une fonction a été implémenté afin de parser ces données sous forme de dictionnaire.

La première grande étape de ce projet est de calculer une matrice de score de similarité entre toutes les paires de séquences grâce à l'algorithme de Needleman & Wunsch.

L'algorithme de Needleman & Wunsch est une méthode de programmation dynamique qui permet, dans un premier temps, de calculer le score d'alignement optimal entre deux séquences. Ce fut la première application de la programmation dynamique pour comparer des séquences biologiques. La première étape de cet algorithme est d'initialiser la première ligne et la première colonne de la matrice de score avec des pénalités de gap, puis calculer le score pour toutes les positions d'acide aminés (en excluant la première ligne et la première colonne) en prenant le score maximal à chaque étape.

Cet algorithme prend en arguments deux séquences protéiques, la matrice BLOSUM62, et une pénalité de gap. Le gap est une valeur de pénalité fixe pour chaque écart, ici la valeur du gap est -5. La matrice

BLOSUM62 est une matrice de similarité/ de substitution. Elle va permettre de définir un score de similarité ou de ressemblance entre deux acides aminés.

Après avoir calculer la matrice de score de toutes les paires de séquences, le but est de calculer la matrice de distances. Cette conversion permet de ne pas avoir de valeurs négatives. Dans la matrice des scores, des valeurs élevées signifiaient que deux séquences étaient proches ; avec la matrice des distances, des valeurs petites indiquent des séquences proches.

Le but à présent est de construire un arbre grâce à la méthode UPGMA (Unweighted Pair Group Method with Arithmetic Mean, Sokal et Michener). UPGMA est issue des méthodes de clusterisation. L'objectif est, par itérations successives, de réduire progressivement la taille de la matrice de distances en la transformant en arbre enraciné, jusqu'à ce qu'il ne reste qu'un seul cluster. Pour cela, l'algorithme recherche dans la matrice de distances les deux clusters ayant la plus petite valeur, les fusionne afin de former un nouveau cluster puis va recalculer la distance entre ce nouveau cluster et les autres clusters. Ces nouvelles distances se calculent en utilisant la formule de la moyenne arithmétique non pondérée des distances entre les séquences des clusters. Les séquences sont ensuite alignées en groupes correspondant à l'ordre de ramification de l'arbre UPGMA.

Arrive maintenant l'étape de l'alignement multiple afin de pouvoir visualiser l'alignement entre toutes les séquences suivant l'ordre de ramification de l'arbre UPGMA. Cette étape se fait grâce à la méthode de Needleman & Wunsch. C'est à partir de la matrice de scores entre deux séquences, que l'étape de tracement des alignements peut se faire. Le but est de retracer le 'chemin' suivant le score maximal des alignements qui donnera l'alignement optimal entre deux séquences. Une fois ce premier alignement effectué, il suffit de prendre la 3ème séquence la plus proche et pour chaque acide aminé, on calcule le score avec chaque acide aminé des séquences précédentes, puis on fait une moyenne. Pour ce faire, il suffit de prendre un vecteur qui calcule la fréquence des acides aminés qu'il rencontre, et calculer le score à chaque position en se référant à la matrice BLOSUM62. Puis comme pour un alignement par pair, on retrace le chemin afin d'afficher les alignements optimaux.

III. Résultats et discussion

Nous allons comparer les résultats que l'on a obtenu expérimentalement et ceux des auteurs. Deux résultats ont été mis en avant dans l'article scientifique : le temps requis pour l'alignement multiple de 10 séquences avec des longueurs de séquences croissantes et le temps des différentes étapes d'alignements pour 90 séquences.

N'ayant pas réussis à faire une fonction qui permet de faire un alignement multiple, mais seulement un alignement par pair, il est difficile de comparer les résultats obtenus avec ceux des auteurs. Néanmoins, j'ai tout de même mesuré le temps d'exécution du script pour 10 séquences protéiques de 400 acides aminés en moyenne (les séquences protéiques sélectionnées sont composées de 110 à 769 acides aminées). Le script s'exécute en 6 à 7 secondes.

En comparaison, pour l'alignement de 10 séquences composées de 400 acides aminés, le temps d'exécution était de 8 minutes. Ce temps supplémentaire vient bien entendu du fait qu'ils ont un alignement multiple et que leurs ordinateurs étaient certainement moins performants que les ordinateurs actuels.

IV. Perspectives

Dans un premier temps, il aurait fallu que je réussisse à implémenter une fonction permettant de faire l'alignement multiple avec la méthode expliquée dans la partie algorithmes.

Afin d'améliorer les résultats, nous pouvons changer la valeur du gap en adaptant la pénalité de gap suivant si on a une ouverture de gap ou une extension de gap. Il serait également intéressant d'ajuster la valeur du gap suivant les acides aminés qui l'entourent avec une pénalité qui diminue si le gap se situe au sein d'acide aminés polaires.

L'algorithme de programmation dynamique de Needleman & Wunsch étant très couteuse en temps et en espace pour des séquences trop longues et/ou trop nombreuses, il est possible d'utiliser d'autres algorithmes comme l'algorithme des k-mers. L'algorithme des k-mers recherche des mots de longueur k ou k-tuple et permet de trouver des sous-séquences (un enchainement d'acides aminés) les plus longues entre deux séquences. Le but est de fractionner chaque séquence et de comparer la longueur de chaque sous-séquences dans les deux séquences afin de trouver la sous-séquence la plus longue. Cela permet de diminuer la complexité algorithmique.

Pour diminuer la complexité de temps pour la construction de l'arbre UPGMA, on peut utiliser la méthode du Neighbour Joining qui va effectuer une recherche séquentielle des voisins en minimisant la longueur totale de l'arbre et produit des arbres enracinés.

Pour la visualisation des résultats, il serait judicieux de mettre un score global afin d'avoir une meilleure visualisation d'ensemble des résultats et d'ajouter des codes couleurs pour mieux visualiser les régions conservées par exemple.