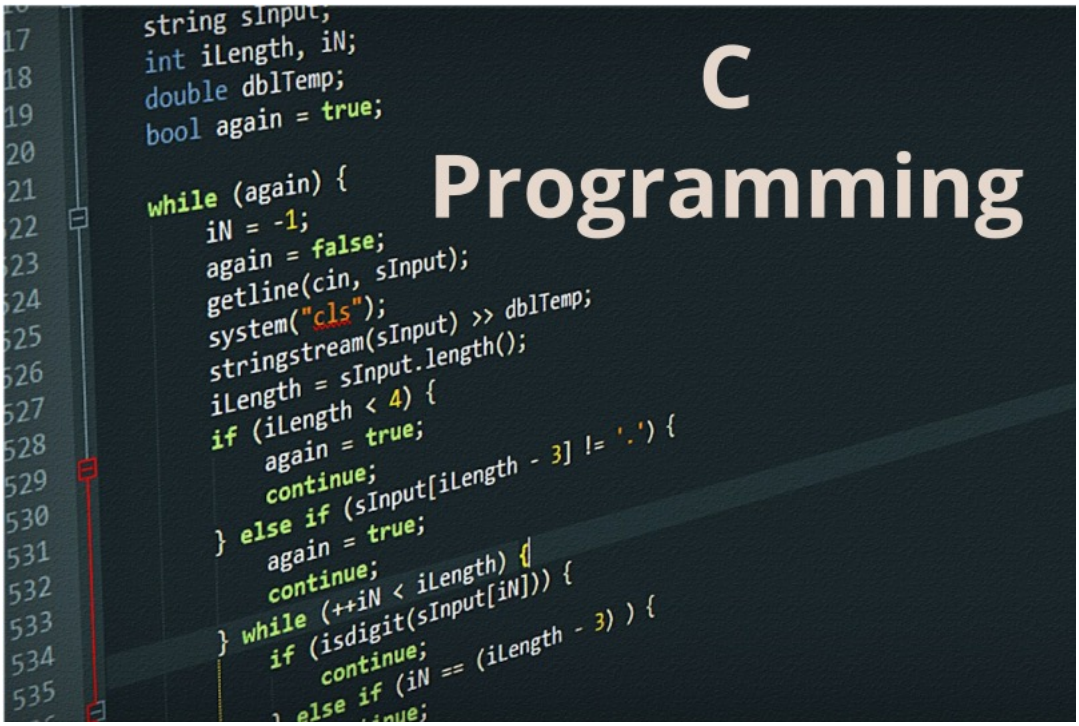
université
PARIS-SACLAY

Master 1 Bio-Informatique et Biostatistique (BIBS)

UE MiNiInfo1 C

Projet- Logiciel d'analyse de séquences



Hammami Siwar
Alouane Aida

Enseignant :
Mme.Chevalier Stephanie

2021 — 2022

Table des matières

1 Introduction 2
2 Expression du besoin 2
3 Fonctionnalités 3
3.1 Utils 3
3.2 Recherche de la séquence codante de taille maximale 3
3.3 Transcription d'une séquence ADN en séquence ARN 3
3.4 Traduction d'une séquence codante en séquence protéique 3
3.5 Calcul du score d'identité entre deux séquences 3
3.6 Calcul du score de similarité de polarité entre deux séquences protéiques . . 4
3.7 Recherche d'une séquence consensus à partir d'un alignement multiple . . 4
4 Conclusion 4

1 Introduction

La Bio-Informatique est la mise en œuvre de méthodes, de concepts ou d'algorithmes éprouvés pour résoudre les problèmes posés par la biologie. Elle nous permet de mettre en œuvre un logiciel d'analyse de séquences génomiques et protéiques eucaryotes, mettant à disposition de l'utilisateur des modules complémentaires pouvant s'enchaîner à la manière d'un pipeline.

Notre projet consiste justement à travailler sur des sous-projets indépendants avec 4 niveaux différents. Le premier consiste au calcul de score d'identité entre deux séquences et faire l'étape de transcription, le second nous permet d'effectuer la traduction et le calcul du score de similarité de polarité entre deux séquences protéiques. Nous procédons ensuite à la recherche d'une séquence consensus à partir d'un alignement multiple et d'une séquence codante de taille maximale et enfin la recherche de la plus grande sous-chaîne de polarité commune à 2 séquences protéiques.

Notre programme repose sur le langage de programmation C. Il s'agit d'un langage compilé qui contient différents types de fichiers distingués par leurs suffixes (.c, .h, .o).

Par conséquent, nous nous retrouvons avec un défi à relever, celui de faire un script qui est capable à la fois de gérer la complexité des informations et d'avoir des sorties de fichier efficaces à la fois sur le plan fonctionnel et ergonomique.

2 Expression du besoin

Notre programme doit être capable de répondre aux besoins spécifiques de l'utilisateur liés à l'analyse séquentielle. Pour cela, il doit pouvoir être capable de traduire puis de transcrire une séquence d'ADN codante en se basant sur les règles spécifiques du code génétique tout en faisant appel à différents modules. Ainsi, à partir d'une séquence nucléotidique le logiciel nous permet de reconnaître la séquence de la protéine potentiellement produite.

De plus, notre programme doit permettre à l'utilisateur de ce dernier de comparer cette séquence à d'autres séquences afin d'étudier leur similarité en se reposant sur un calcul de score d'identité et de similarité.

Finalement, grâce aux modules de ce programme, on doit pouvoir identifier les régions les plus conservées entre plusieurs protéines issues de la même famille. Le but étant de pouvoir identifier quelles sont les zones de la protéine issue de la séquence d'ADN, qui sont essentielles à la fonction protéique.

3 Fonctionnalités

3.1 Utils

Le fichier utils.h fait référence à utils.c. Ce dernier contient les fonctions utiles à tous les modules et qui pourront être appelées depuis chacun d'entre eux. Ces fonctions sont :

-void get_path_from_user(char* path_input) : le stockage du path des fichiers choisis par l'utilisateur
-void extract_sequence(const char* path_input, char* sequence) : extraction des séquences du fichier choisis par l'utilisateur pour pouvoir effectuer nos modules
-void save_sequence(const char* path_output, char* sequence) : sauvegarde du résultat dans un fichier tout en laissant à l'utilisateur de choisir son propre path et le nom du fichier de sortie

3.2 Recherche de la séquence codante de taille maximale

Dans ce module est effectué avec la fonction "void seqCodanteMain()". Nous avons commencé tout d'abord à travailler dans le sens normal en parcourant au début la séquence caractère par caractère jusqu'au point d'arrivée le codant START, puis nous avons continuer pour le reste de la séquence trois par trois jusqu'à ce que nous sommes arrivée à trouver un codon STOP. Ensuite pour le brin anti-sens, nous avons inversé la séquence puis remplacé les "T" par "A" et "C" par "G" et inversement, puis nous avons refait ce que nous avons fait précédemment dans le sens normal. La récupération de la plus grande séquence s'effectue dans "seqPlusLongue". Finalement nous l'avons enregistré dans un fichier grâce à la fonction "save_sequence(path_output,seqPlusLongue)" du fichier utils.c

3.3 Transcription d'une séquence ADN en séquence ARN

La transcription permet de transformer notre séquence ADN en séquence ARN, ce qui explique la transformation de T par U tout en vérifiant que la séquence commence bien par un ATG, cette fonction est bien appelée "void transcriptionMain ()".

3.4 Traduction d'une séquence codante en séquence protéique

Le module de traduction permet de récupérer en input la sortie du fichier de transcription. En effet ce dernier permet de parcourir la séquence caractère par caractère et chaque trois caractères est représenté par une protéine, cette partie est faite avec "char prot(char car1 , char car2 , char car3)" Ce module est bien effectué avec la fonction "void traducionMain ()".

3.5 Calcul du score d'identité entre deux séquences

"calculScoreMain()" permet la récupération de deux séquences, de tester par la suite la longueur de ces deux séquences pour pouvoir trouver le résultat du score d'identité tout en les comparant entre elles.

3.6 Calcul du score de similarité de polarité entre deux séquences protéiques

Le fichier calculScorePolarité permet l'affichage du score, avec void calculScorePolariteMain(). Ce dernier est trouvé par la répartition des protéines hydrophiles avec un score de 0, les protéines hydrophobes ayant un score de 1 et les différents n'ont pas de score donc un -. Ainsi avec la récupération de deux séquences nous devons toujours vérifier l'égalité en testant la longueur des deux séquences.

3.7 Recherche d'une séquence consensus à partir d'un alignement multiple

La recherche d'une séquence consensus à partir d'un alignement multiple fonctionne avec la fonction principale "void alignementMain()", elle fait appelle à deux autres fonctions, "void getMaxRepeatingElement()" qui permet de vérifier et de compter la fréquence de chaque élément. Si elle est supérieure au nombre maximal d'éléments que nous avons trouvé jusqu'à présent, elle sera mise à jour,et "void creationSeqConsensus". De plus, nous avons pu effectuer une vérification de la séquence qui contient bien ATCG et - avec "bool verifSeqADN(char sequence[10000])".

4 Conclusion

Finalement, l'élaboration de ce logiciel d'analyse de séquences génomiques et protéiques eucaryotes n'a pas été une tâche très facile dès le départ. Dans le sens où nous avions tellement d'idées et d'informations pour procéder mais pas de plan spécifique et bien structuré.

Ainsi, il a été nécessaire de nous organiser, de faire des fiches, filtrer nos idées et nous mettre à la place de l'utilisateur afin de pouvoir mettre en place les différents modules complémentaires répondant à la demande de l'utilisateur et de réaliser l'analyse des séquences tout en gardant une utilisation relativement agréable et simple à comprendre et à visualiser. Nous avons alors été capables de proposer un programme qui permet d'interagir avec l'utilisateur et de lui laisser la liberté de choisir un certain nombre de paramètres.

Le côté ergonomique de notre logiciel se trouve dans l'utilisation de différentes couleurs et différentes lettres, par exemple l'affichage des résultats de succès en bleu et d'erreurs en rouge. Notre groupe s'est organisé pour que chacune de nous fasse des modules ou des fonctions séparément pour ensuite fusionner le tout dans un seul script séparé en différents fichiers tout en utilisant GitHub. Cela nous a seulement permis de gagner en efficacité, et à obliger chacune de nous à comprendre l'intégralité du code afin qu'elle puisse faire fonctionner sa partie dans le script commun.

Bien sûr, nous avons rencontré plusieurs difficultés pendant l'élaboration de notre projet, parmi elles on retrouve :

- le stockage du path des fichiers choisis par l'utilisateur
- La recherche d'occurance
- Le stockage de la séquence codante de taille maximale

Nous avons également pensé à plusieurs idées que l'on aurait voulu ajouter au programme mais aussi quelques erreurs que l'on aurait voulu corriger mais malheureusement nous n'avons pas eu le temps de les ajouter et de les corriger.

Par exemple, nous aurions souhaité développer notre projet dans une interface pour que ce dernier soit plus visible. Également, il aurait été préférable de développer le niveau 4 qui consiste à rechercher la plus grande sous-chaîne de polarité commune à 2 séquences protéiques.

Pour conclure, nous avons été dans l'obligation de prendre certaines décisions durant toute la durée du projet pour atteindre notre objectif sans trop dévier de ce dernier. Par exemple, nous avons dû travailler par niveau et bien se répartir le travail et essayer de partir directement au but.