Introduction à la Bioinformatique

Projet Python

Par : **Ines MAHOUT** et **Abdelhamid STA**

INTRODUCTION ET OBJECTIF

La bio-informatique est une discipline émergente de la recherche qui se place à l'interface de la biologie et de l'informatique dans le but de résoudre un problème scientifique posé par la biologie.

Ce projet bio-informatique consiste donc en la réalisation de plusieurs fonctions via python, ayant pour objectif de **tester si certaines séquences de transcrits peuvent être la cible d'un microARN**, et ce, en identifiant les différentes régions de fixation de notre microARN.

Nous allons dans un premier temps définir ce qu'est un micro-ARN et identifier pourquoi il représente un grand intérêt dans la recherche thérapeutique, à l'ère de la médecine personnalisée. Ensuite nous allons rentrer dans le cœur de notre projet en expliquant nos différentes fonctions et les stratégies mises en place afin d'optimiser notre projet. Enfin nous discuterons des différentes approches personnelles et nos remarques concernant ce projet.

I) DEFINITION ET ROLE BIOLOGIQUE

En effet, les **microARN** sont de petits ARN (petit fragment simple-brin de 21 à 24 nucléotides de long) pouvant réguler l'expression des gènes au niveau post-transcriptionnel, en s'appariant à des ARN messagers portant une séquence homologue.

Deux mécanismes distincts sont impliqués en fonction du degré de complémentarité entre la séquence du miARN et celle de l'ARNm cible :

1) Lorsque la complémentarité est parfaite (prédominant chez les végétaux), le complexe RISC assurant l'interaction entre le miARN et l'ARNm permet l'action de ribonucléases capables de cliver, et donc de dégrader, la molécule d'ARNm.
2) Lorsque la complémentarité est partielle, ce qui est prépondérant chez les mammifères, l'ARNm n'est pas clivé, mais la liaison du miARN aux régions non traduites en 3' (séquences UTR [untranslated regions]) est à l'origine d'une inhibition de la traduction en protéine.

Dans ces deux cas, la conséquence est une **extinction de l'expression du gène correspondant**.

Il a aussi été montré que les microARN peuvent directement méthyler l'ADN afin d'éteindre des gènes.

Les microARN forment une des grandes voies de régulation de l'expression des gènes, ils possèdent donc un intérêt thérapeutique majeur, notamment en tant que biomarqueurs. Leurs dérégulations étant impliquées dans certaines pathologies telles que le cancer ou la maladie d'Alzheimer.

II) STRATEGIE ET PROGRAMME

Pour répondre aux besoins du biologiste, nous avons créé un programme en nous basant sur les propriétés de fixation d'un microARN :

- Le microARN se fixe dans la région 3'UTR du transcrit cible
- Le microARN reconnaît une séquence sur le transcrit cible qui est complémentaire inverse de la région appelée *seed* allant du 2ème au 7ème nucléotide du microARN

Nous disposons, pour cela, de plusieurs outils pour construire un programme robuste tels que : les fiches *Genbank*, les séquences des transcrits ainsi que les séquences des microARN au format *Fasta*.

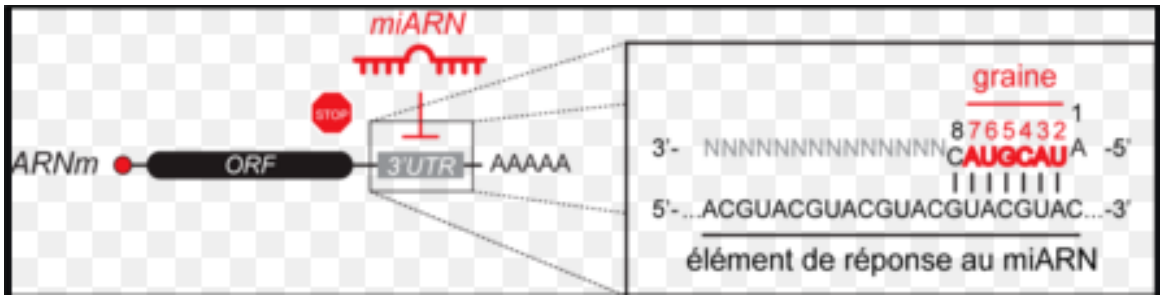
Pour pouvoir ouvrir et lire le fichier, nous avons créé la fonction « ***readFasta*** », qui va le lire et renvoyer séparément le ***header***, correspondant à la première ligne, et la séquence, à partir de la deuxième ligne. Nous avons aussi une autre fonction « ***write_on_file*** » pour enregistrer nos résultats sous l'extension voulue.

Notre programme se construit en plusieurs parties :

Partie 1 : Séquence complémentaire

➔ *Créé par Ines*

Comme dit précedemment, les miARN s'hybrident principalement aux régions 3'UTR des ARNm, **par complémentarité parfaite de la séquence "seed"** qui s'étend sur 6-8 nucléotides à l'extrémité 5' du miARN (à partir du 2ème nucléotide en 5').



Le but de cette première partie va donc être de créer un programme qui renvoie la **séquence complémentaire inverse d'une séquence nucléotidique fournie au format fasta**.

(Fichier test : seed.fas)

Pour cela notre fonction « ***reverse_complement*** » va prendre en entrée une séquence nucléique, va parcourir cette séquence et renvoyer pour chaque nucléotide trouvé son complémentaire :

- La lecture du nucléotide **A** va renvoyer **T**
- La lecture du nucléotide **C** va renvoyer **G**
- La lecture du nucléotide **G** va renvoyer **C**
- La lecture du nucléotide **T** va renvoyer **A**

- Pour trouver l'inverse de notre séquence nous avons crée une variable de type « ***str*** » vide, et qui va à mesure de la lecture récupérer le nucléotide complémentaire et **l'ajouter à la gauche** de la séquence qu'il va renvoyer. Le premier élément trouvé sera à terme le dernier de notre variable.

Soit pour une séquence ATTG, le complémentaire va être, dans l'ordre de lecture :

1. T
2. AT
3. AAT
4. CAAT
5. CAAT

- Nous avons ensuite établi diverses possibilités d'un **renvoie d'un message d'erreur** si :

- Le fichier n'est pas disponible
- Le fichier n'est pas dans le bon répertoire
- L'entrée n'est pas un string
- La séquence contient des espaces
- La séquence n'est pas de type nucléotidique

Partie 2 : Récupération de la Séquence 3'UTR via une fiche genbank

➔ *Créé par Hamid*

C'est en se fixant sur la région 3'UTR des ARNm correspondants que les miARN régulent l'expression génique.

La deuxième partie est donc un programme qui **récupère la séquence de la région 3'UTR** d'un transcrit grâce à **l'annotation de la position de la région codante (CDS)** dans sa fiche Genbank et à sa séquence au format fasta.
(Fichiers test : smad1.gb, smad1.fas)

- Cette partie s'est déroulée en deux étapes :

1. Nous avons d'abord **récupéré la partie 3'UTR** de la séquence d'un fichier *Genbank* via la fonction « ***recherche_positions*** ».

Comme les fichiers Genbank possèdent la même structure, nous avons :
- Identifié le sigle *CDS* correspondant à la partie codante
- Récupéré la **valeur fin du CDS +1** correspondant au **début de la partie 3'UTR**
- Récupéré la taille totale de la séquence définie par le sigle *locus*.

2. Ensuite il a suffi de parcourir le fichier fasta et de **récupérer la séquence entre nos deux valeurs**, soit : [***valeur_de fin_CDS*** : ***taille_totale_sequence***].

- Concernant les **erreurs possibles** :

Nous avons pris en compte si le fichier Genbank **ne correspond pas au fichier *Fasta***, via l'implémentation de la fonction « ***comparaison_version*** » qui permet de récupérer les numéros de version des deux fichiers. En les comparant, on peut identifier si les fichiers correspondent à la même séquence.

De plus, si l'entrée correspond à un fichier valide mais **pas à un fichier Genbank**, il y a renvoi d'un ***IndexError***.

Partie 3 : Recherche de motif exact

➔ *Créé par Ines*

- Après obtention de notre séquence complémentaire à la région *seed* et de la séquence 3' UTR, on va écrire un programme qui teste **si une séquence courte (motif exact) est présente dans une autre séquence** et précisera la/les **position(s) de chaque début de match**. (Fichiers test : motif.fas, 3utr.fas)

Nous allons donc, dans cette partie, parcourir la séquence et utiliser un curseur qui va identifier la région similaire à notre séquence.
Une fois que nous avons trouvé le motif dans sa globalité, nous allons renvoyer la position du premier hit correspondant au motif.

- Soit pour une **séquence AABB** et de **motif BB**, le résultat va être **([2], 1)** correspondant respectivement à la position et le nombre de motif trouvé.

Partie 4 : Recherche de sites cibles d'un microARN dans un transcrit

➔ *Testé par Hamid*

Ensuite, nous allons tester si le microARN ***mmu-mir-30a*** peut cibler le transcrit de ***smad1***. (Fichiers test : mmu-mir-30a.fas, smad1.gb, smad1.fas)

La recherche de site cibles va donc représenter la phase finale de notre projet, via la réutilisation des précédentes fonctions afin d'identifier les sites cibles d'un micro ARN.

- Nous savons tout d'abord que le micro-ARN se fixe sur la région 3'UTR, qui sera donc identifié via la fonction « ***recherche_positions*** », ensuite nous allons faire la **reverse complémentaire** de cette séquence, puis récupérer la région *seed* de notre micro ARN soit [**1 :7],** et enfin réaliser une **recherche de motif** de la région seed, sur la séquence 3'UTR reverse complémentaire.

- Concernant la **gestion des erreurs**, on va prendre en compte si l'utilisateur ne rentre pas le bon fichier fasta correspondant pour localiser la 3'UTR, si le fichier est bien de type GB (contient une *CDS*) , ou si le fichier n'est pas dans le bon répertoire.

Partie 5 : Récupération sur internet des séquences

➔ *Par Ines et Hamid*

Enfin nous allons récupérer des séquences depuis internet via la fonction « ***download_sequence*** » qui va pouvoir ouvrir les fichiers de type ***mirbase*** ainsi que *Genbank* et *Fasta*.

La gestion des erreurs nous permet d'identifier si l'entrée est valide ou non.

III) APPROCHE PERSONNEL

Ce code a donc été organisé en 5 parties et contient plusieurs petits programmes permettant au biologiste de tester si certaines séquences de transcrits peuvent être la cible d'un microARN.

Il a été optimisé au maximum, cependant, une difficulté a été de s'adapter au format de certains fichiers. Notamment ceux de la partie 5 concernant la récupération sur internet des séquences.

De plus, il serait judicieux de créer une ***interface graphique***. En effet, l'interface graphique désigne la manière dont est présenté un logiciel à l'écran pour l'utilisateur. C'est le positionnement des éléments : menus, boutons, fonctionnalités dans la fenêtre. Une interface graphique bien conçue est ergonomique et intuitive afin que l'utilisateur la comprenne tout de suite.