College Science and Technology

université de BORDEAUX

Project - Alone in the dark Development of a genetic analysis tool : ORF finder

Authors:

BLASQUIZ Julie : *julie.blasquiz@etu.u-bordeaux.fr*

JUNG Frédéric : *frederic.jung@etu.u-bordeaux.fr*

THOUVENIN Arthur : *arthur.thouvenin@etu.u-bordeaux.fr*

Bordeaux

29-11-2017

LaTeX

Contents

1 Introduction

En Génétique, la source de données sur laquelle repose tout projet provient d'un ou plusieurs génomes d'espèce spécifique. Dans ces génomes, l'intégralité de l'ADN contenu n'est pas codant; Après les étapes de translation et traduction seules les parties codantes ont la probabilité d'être traduites en protéines. Toute étude génétique s'appuie sur les résultats d'obtention de ces parties codantes et plus précisément sur la détection des ORFs pour notamment concevoir des amorces nécessaires pour des réalisations de PCR ou du séquençage d'ADN.

Un ORF (Open Reading Frame) est un des trois cadres de lecture possible dans lequel un ARNm peut être potentiellement traduit en protéine. Dans l'étude de gène, un ORF est caractérisé par la séquence de nucléotides qui, après être transcrit en ARNm, se traduit par une série de codons qui n'est interrompu par un codon stop en fin de séquence.

Quelques outils bio-informatiques permettent de trouver tous les ORFs d'une séquence nucléique donnée en entrée. NCBI-ORFfinder en est un bon exemple; ce programme retourne l'intégralité des ORFs d'une séquence spécifiée avec leurs traductions en séquences protéiques. Dans ce programme, des paramètres sont à définir avant exécution comme la longueur minimale exigée pour la détection d'un ORF, le code génétique ou encore les codons start et stop à prendre en compte.

ORF-investigator en est un autre. Celui-ci repose sur deux algorithmes particuliers : l'algorithme de Needleman et celui de Wunsch qui permettent tout deux un alignement global de séquence nucléique donnant ainsi, en plus des séquences ORF trouvées, la localisation précise des ORFs détectés.

Il existe d'autres programmes pour trouver les ORFs comme ORFm [1] qui permet d'identifier très rapidement les ORFs. Il prend en entrée un séquence FASTA et en retourne les ORFs en sortie en format FASTA avec la position de départ, la taille de l'ORF (en nombre de triplet) et le numéro de l'ORF. Le threshold est réglé par défaut à 96pb. Il utilise l'Algorithme d'Aho-Corasick [3] qui permet de rechercher un motif dans du texte en lisant les lettres du texte une par une.

L'obtention d'ORFs à pour objectif principal la détection de gènes. Les ORFs obtenus peuvent être des gènes ou non. Un gène s'identifie par la présence d'une séquence spécifique appelée promoteur. La recherche d'ORFs doit donc être complétée par la recherche de promoteur afin de permettre la détection des gènes.

Dans ce projet, nous essayons de prédir les différents gènes d'un gènome

complet de la bactérie *Mycoplasma mycoides*. Pour cela, nous créons notre propre outil bio-informatique permettant la détection d'ORFs à l'aide du langage de programmation Python.

2 Matériel et Méthode

La séquence que nous avons à étudier correspond au génome de *Mycoplasma mycoides subsp. capri* (NZ_CP012387) qui est une bactérie associée à la maladie respiratoire de la chèvre avec un taux élevé de mortalité dans les troupeaux[4]. La taille du génome complet est de 1 085 764pb. La séquence du génome étudiée est issue de la banque de données du NCBI (National Cancer for Biology Information).

Lors de la traduction, les codons des ARNm sont traduits en acides aminés. Cette traduction s'effectue en se basant sur un code génétique dit standard (Figure : 1). La traduction commence à la lecture d'un codon start, ATG, et se termine à la lecture d'un codon stop ,TAA, TAG ou encore TGA. Certaines espèces ne respectent cependant pas parfaitement ce code génétique. Notre code prend en considération les modifications du code génétique qui s'applique à l'éspèce étudiée.

Le langage de programmation utilisé pour ce projet est Python (version 2.7.13). On peut lire notre fichier fasta pour y récupérer la séquence. Un autre fichier, nommé myBio.py nous permet de faire appel à une bibliothèque de fonctions, préalablement réalisée, qui nous permettent d'identifier les ORFs. La figure 2 représente de façon schématique l'agencement du projet que nous avons réalisé avec les diverses fonctionnalitées.

Tout d'abord nous avons établit une phase d'initialisation avec le START, c'est grâce à cette étape que l'on définit ce sur quoi nous allons travailler. Deux choix s'offre à l'utilisateur:

- Charger un fichier .csv formaté selon une précédente utilisation
- Ou donner au programme une séquence FASTA qui sera alors analysée de façon à trouver les ORFs

Cette initialisation nous permet de définir la variable avec laquelle nous allons travailler tout au long du programme, c'est à dire une liste contenant des dictionnaires. Ces dictionnaires contiennent toute les informations des ORFs trouvés dans la séquence.

Par la suite c'est une autre fonction qui va gérer les commandes rentrée par l'utilisateur, cette fonction fera appelle à toutes les fonctions nécessaires correspondant aux demandes de l'utilisateur. Ces fonctions feront souvent appel à notre liste de dictionnaires permettant leurs bonnes exécutions.

De plus un système de sauvegarde à été implémenter pour conserver les listes d'ORFs au format CSV et pouvoir les recharger pour les comparer avec des fichiers FlatFile eux mêmes sont lus par le fichier pour être utilisés.

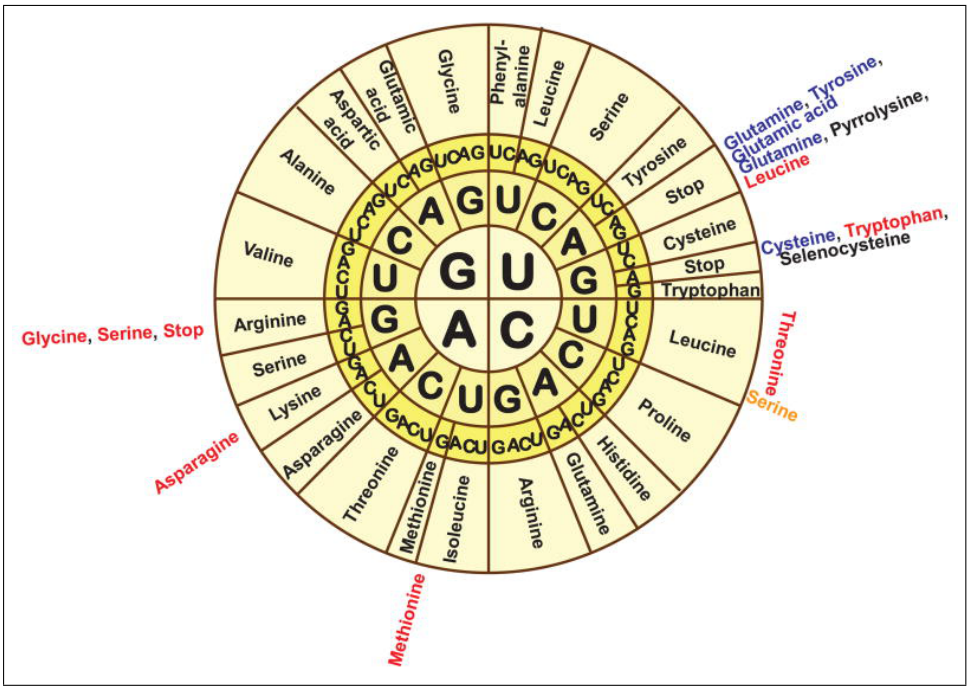

Figure 1: Code génétique avec ses quelques possibilités de variations

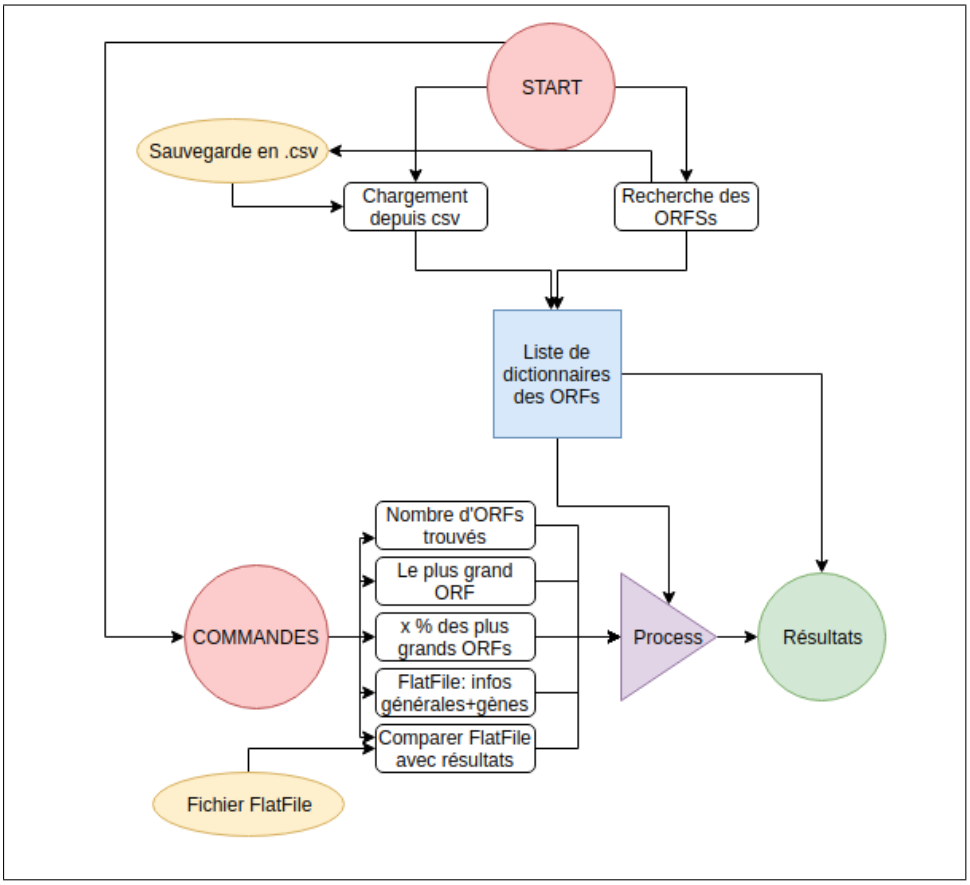

Figure 2: Représentation schématique des fonctionnalités du projet

2.1 Trouver les ORFs

Nous avons écrit une fonction pour détecter les ORFs dans notre séquence. Il faut que l'algorithme soit capable, à partir de la séquence donné en entrée, d'avancer de nucléotide en nucléotide jusqu'à détecter un codon START. Ensuite il lit les triplets de ce même cadre ouvert de lecture jusqu'à détecter un codon STOP (Figure : 3). Si il trouve un codon stop, on est bien en présence d'un ORF, une fonction permet de déterminer la séquence de cet ORF et la stock dans un dictionnaire ainsi que diverses informations comme la position du codon start, sa taille etc.... Et on continue d'avancer jusqu'à avoir trouvé tous les ORFs de la séquence. L'opération sera répétée pour les cadres anti-sens.

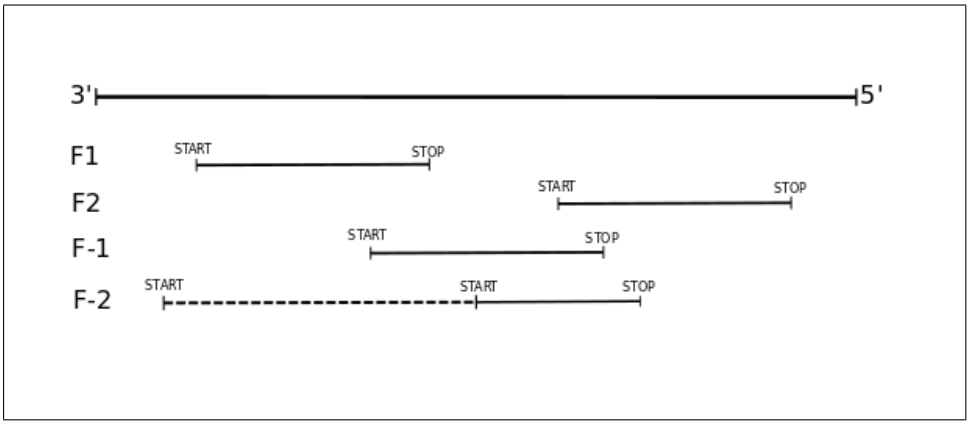

Figure 3: Représentation schématisée des ORFs

Nous avons rencontré deux grands types de problèmes que nous avons réussi à régler: Nous utilisons un système de boucle qui regarde les lettres de la séquence une à une et qui s'arrête lorsqu'il rencontre un START. Il rentre alors dans une deuxième boucle qui scan la séquence pour lui trouver un codon STOP sur le même cadre de lecture. Il a fallu faire des ajustements au script pour que la recherche de ce codon STOP commence après le codon START.

Le deuxième problème est représenté sur la Figure 3 par le trait en pointillet. Il est possible que plusieurs codons start se trouvent sur le même cadre ouvert de lecture. Le programme parcout la séquence jusqu'à trouver un codon STOP. Les codons START qui se trouvent sur le même cadre ouvert de lecture et qui on le même codons STOP ne doivent pas être pris en compte. Seul le premier est conservé. Le programme parcourt donc la séquence frame par frame en ne conservant pas les ORF à l'intérieur d'un autre.

Notre programme prend aussi en compte les ORF sur le brin complémentaire. Un fonction *reverse* nous permet de déterminer la séquence complémentaire. Ainsi, on peut chercher les ORFs de ce brin de la même manière que le brin matrice.

Il a fallu aussi déterminer un seuil (threshold) pour la taille de l'ORF. Si l'ORF est trop petit on ne le prend pas en compte, c'est à dire qu'on ne l'ajoute pas à la liste des ORFs trouvés. Cinq valeurs de seuils sont prédéterminés par une des fonctions du programme : pas de seuil, un seuil de 90pb, 210pb, 300pb et enfin 420pb. L'utilisateur du programme est amené à selectionner l'un de ces seuils proposés.

2.2 Le fichier CSV

Un fichier CSV est un fichier tableur au format texte. C'est à dire qu'il peut être utilisé facilement par un certain nombre de programme et permet également d'avoir une meilleure visibilitée sur les résultats. C'est la finalité du projet, c'est à dire que ce fichier CSV va contenir toutes les informations concernant les ORFs trouvés (taille, position, séquence etc...). Ce genre de fichier peut être utilisé par des logiciels comme Rstudio, ce qui nous permet de faire des statistiques mais peut être aussi utilisé par des langages de programmation comme LaTeXpar exemple car ce format de fichier répond à des conventions bien précises et est donc facilement exploitable. La fonction *readCSV* de notre programme permet de prendre un fichier CSV en entrée et de le convertir dans notre programme en liste de dictionnaire qui peut être exploité pour obtenir des informations sur les ORFs stockés dans ce fichier.

L'ensemble des fonctions du script ainsi que leurs interactions entre elles sont résumées dans la figure ci-dessous.

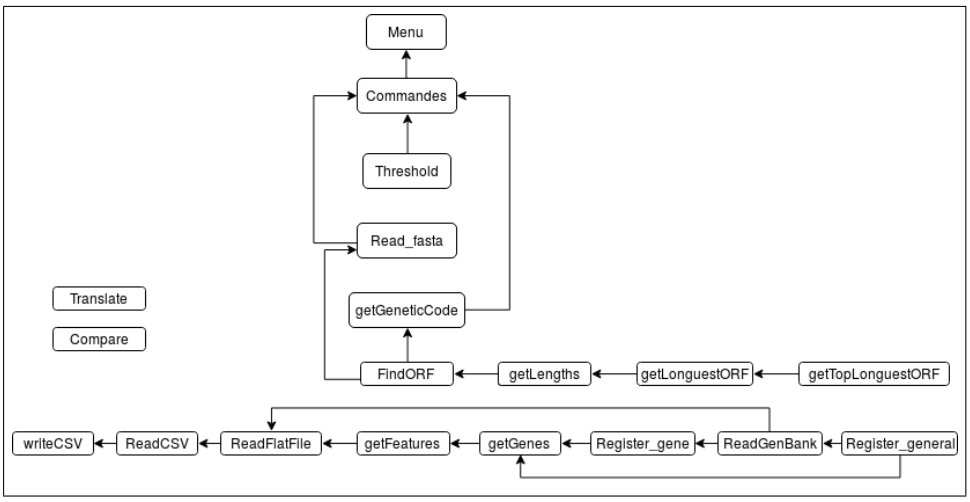

Figure 4: Représentation schématisée des fonctions du script ainsi que leurs interactions entre elles

Table 1: Resultats du programme

	Nombre d'ORFS détecté	**Longueurs (en nc)**
Pas de seuil	58052	3 à 603
90 bp	4088	90 à 603
210 bp	135	210 à 603
300 bp	16	303 à 603
420 bp	2	426 à 603

Table 2: Resultats du programme ORFfinder du NCBI

	Nombre d'ORFS détecté	**Longueurs**
30 bp	456	33 à 3084
75 bp	184	78 à 3084
150 bp	85	153 à 3084
300 bp	38	312 à 3084
600 bp	21	603 à 3084

3 Résultats

Le programme préalablement décrit est exécuté pour chacun des seuils suivants : pas de seuil, 90bp, 210bp, 300bp et 400bp. Les résultats obtenus sont résumés dans le tableau Table1.

Pour une valeur seuil de 420 bp 2 ORFs ont été trouvé par le programme; 16 ORFs ont été trouvé pour une valeur seuil de 300bp; 135 pour un seuil de 210 bp; 4088 pour un seuil de 90 bp et enfin 58052 ORF lorsqu'il n'y a pas de seuil. En comparant le nombre d'ORF trouvé avec l'outil ORFfinder du NCBI et le nombre d'ORF détecté par notre programme pour un même seuil à 300 bp, on remarque que l'outil du NCBI en détecte plus (38 contre 16 avec notre programme) et de longueur plus importante (jusqu'a 3084 bases contre 603 bases avec notre outil) La banque de données GenBank du NCBI recense 942 gènes pour le gènome de la bactérie *Mycoplasma mycoides subsp. capri*. Le nombre d'ORF détecté étant le plus proche de ce nombre correspond aux 135 ORFs obtenus pour une valeur seuil de 210 bp. Ce nombre est très largement inférieur au nombre de gènes attendus dans ce gènome. On peut etre amené à penser que les 135 ORFs détectés font partie des 942 gènes attendu et qu'ainsi, 807 gènes n'ont pas été retenu par le programme. Ces gènes non retenus peuvent être des gènes dit non codants, non contenu dans des ORFs, mais qui sont cependant recensés par NCBI.

4 Discussion et Conclusion

Le programme que nous utilisons nous permet d'identifier les ORFs d'une séquence et de donner leurs caractéristiques. Les informations sont mobilisables dans un fichier CSV. Les 135 ORFs détectés par le programme sont suceptibles d'être des gènes. Le programme à donc une bonne spécificité de reconnaissance des ORF. La valeur seuil du programme peut être cependant ajustée de sorte de se rapprocher du nombre d'ORF détecté par l'outil du NCBI.

Par manque de temps nous n'avons pas réussi à optimiser correctement le code pour gagner du temps d'exécution du programme, en effet pour une machine avec 8Go de RAM, un processeur Intel Core i7-7500U CPU @ 2.70GHz x 4, une carte graphique Intel HD Graphics 620 (Kaby Lake GT2), avec Ubuntu 17.10-64bit et 61.9Go de disque dur, divers tests ont été réalisé : Table 3

Table 3: Resultats des tests de performance en secondes

	Sans activités parallèles	**Avec activités parallèles**
Lecture séquence	10	15
Analyse des ORFs		
treshold 150	114	330
treshold 0	600	-
treshold 90	300	-

On peut ici observer que les délais reste relativement correctes malgrès de possibles améliorations.Si l'on compare par exemple au find ORF de NCBI qui met 10 secondes environ à trouver les ORFs avec un threshold de 150.

De plus avec le manque de temps nous n'avons pu tester la fonction compare(), qui permet de comparer des gènes de GenBank avec les ORFs trouvés par notre programme.

Nous aurions également eu plus de temps nous aurions pu mettre en place une gestion d'erreur qui n'a pu prendre que la forme de la fonction error() dans notre script mais que nous n'avons pas eu le temps d'incorporer.

Ainsi les nombreuses différences avec des outils utilisés partout dans le monde comme le find ORF de NCBI, peuvent s'expliquer de différentes façons, peut être est ce la cause du script en lui même, pas assez optimisé, ou alors à notre amanque de temps pour aboutir à un projet aboutit. Cependant nous nous orientons plus vers la différence d'algorithme avec les programmes professionels, en effet ceux-ci sont beaucoup plus complexes et font référence à des notions de statistiques qui ne sont pas implémenter dans notre script, ces

outils utilise également les bases de données à leur disposition afin d'établir des stratégies prédictives.

On peut alors finir ce projet avec des remerciements pour monsieur Taveau et madame Thebault pour leur précieux conseils durant ce projet

5 Source

- `https://docs.python.org/2/c-api/index.html`
- `https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi`
- `https://www.ncbi.nlm.nih.gov/orffinder/`
- `http://emboss.bioinformatics.nl/cgi-bin/emboss/getorf`
- `http://emboss.bioinformatics.nl/cgi-bin/emboss/help/getorf`
- `http://www.bioinformatics.org/sms2/orf/_find.html`
- `http://www.geneinfinity.org/index.html?er=404`

References

[1] Ben J. Woodcroft, Joel A. Boyd, Gene W. Tyson. OrfM: a fast open reading frame predictor for metagenomic data. *Bioinformatics* , Volume 32, Issue 17, 1 September 2016, Pages 2702–2703. 03 May 2016.

[2] Irene T. Rombel, Kathryn F. Sykes, Simon Rayner, Stephen Albert Johnston. ORF-FINDER: a vector for high-throughput gene identification. *Elsevier*; Received 27 July 2001; accepted 10 November 2001; Gene 282 (2002) 33–41

[3] Saima Hasib, Mahak Motwani, Amit Saxena; Importance of Aho-Corasick String Matching Algorithm in Real World Applications (IJCSIT) *International Journal of Computer Science and Information Technologies*, Vol. 4 (3) , 2013, 467-469.

[4] Laura Hernandez, Jose Lopez,corresponding author Marcel St-Jacques, Lourdes Ontiveros, Jorge Acosta, and Katherine Handel. Mycoplasma mycoides subsp. capri associated with goat respiratory disease and high flock mortality. *Can Vet J.* 2006 Apr; 47(4): 366–369.

[5] Alexey V. Lobanov, Anton A. Turanov,Dolph L. Hatfield, and Vadim N.Gladyshev. Dual functions of codons in the genetic code. *Crit Rev Biochem Mol Biol* 2010 Feb; 45: 257–265.