Projet Bioinformatique

Analyse de la régulation transcriptionnelle du gène *uspA* chez *Escherichia coli*

Fahy Alexis
Licence 2 S3

Table des matières

1 Introduction

J'ai souhaité réaliser ce projet afin de mettre en pratique des connaissances récemment acquises lors d'un cours d'introduction à la bioinformatique. Le projet est parti d'une question très simple : si une protéine est fortement exprimée chez un organisme, est-ce dû à de nombreuses répétitions du gène dans le génome ou plutôt à des mécanismes de régulation transcriptionnelle ou traductionnelle ?

Bien que des outils existent pour répondre à cette question dans le cadre du gène étudié ici, *uspA* [3], j'ai choisi de créer un programme afin d'analyser le génome sans passer par ces outils dans un premier temps.

Le projet a été implémenté en langage C pour bénéficier de son approche bas niveau, de son efficacité en termes de calculs et de gestion des ressources. J'ai également utilisé de nombreuses ressources, comme les banques de données RefSeq du NCBI ou encore l'outil de criblage de banques Blastn, utilisé plus tard après l'analyse préliminaire. Ce projet se veut le plus rigoureux possible, mais des erreurs ont sûrement été commises et je m'en excuse par avance. Il ne s'agit pas d'un projet de recherche, mais bien d'un projet personnel visant à mettre en pratique les cours reçus jusque-là.

FIGURE 1 – Image de *Escherichia coli* [4].

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

FIGURE 2 – Comparaison de différents langages de programmation [5].

2 Démarche

2.1 Objet de l'étude

L'organisme choisi pour cette étude est la bactérie ***Escherichia coli* K12 substr. MG1655**, qui présente l'avantage d'être un modèle très étudié. Il existe donc de nombreuses données disponibles à son sujet. J'ai d'abord téléchargé son génome complet [1] depuis la base de données **RefSeq du NCBI**.

Ensuite, l'idée était de trouver un gène d'intérêt à étudier. Pour cela, j'ai utilisé la base de données **PaxDb 5.0** [2], qui fournit des informations sur l'abondance des protéines. Cette approche m'a permis de sélectionner le gène *uspA* [3], située sur le brin positif, qui code pour la protéine **UspA** [7] (*Universal Stress Protein A*).

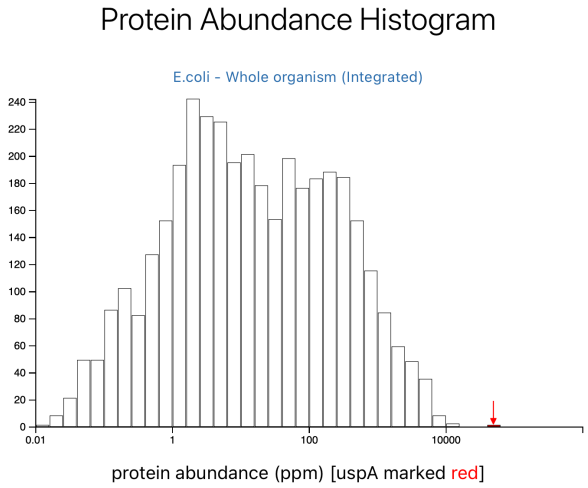

FIGURE 3 – Histogramme de l'abondance des protéines dans *E. coli*, avec *UspA* marqué en rouge [2].

FIGURE 4 – Structure 3D de la protéine *UspA* [6].

2.2 Présence répétée du gène ou forte activité transcriptionnelle ?

Pour répondre à cette question, j'ai commencé par créer un programme [12] permettant de rechercher le gène d'intérêt (la séquence CDS) dans le génome, tout en indiquant sa position et son taux d'identité par rapport à la séquence de référence. J'ai utilisé un taux d'identité minimal de **80 %**. Le programme est capable de repérer et d'indiquer si le gène d'intérêt est présent en plusieurs exemplaires.

```
================================================================================
   Recherche du gène dans la séquence réelle d'E. coli
================================================================================

Gène trouvé à la position 3640111 avec une identité de 100.00%
```

FIGURE 5 – Capture d'écran du résultat du programme

Le programme a trouvé le gène *uspA* à la position **3 640 111** sur le génome d'*E. coli* K12 avec une identité de **100 %**. Cela montre que le gène est présent une seule fois sur le brin positif et qu'il est potentiellement soumis à des mécanismes de régulation transcriptionnelle ou traductionnelle plus complexes qu'une simple présence en plusieurs exemplaires pour expliquer son expression élevée.

2.3 Analyse des boites consensus

Par la suite, l'objectif était de vérifier la présence de boîtes consensus en amont du gène. La distance fixée arbitrairement pour délimiter la zone d'étude fut de **200 paires de bases** en amont du gène. En effet, à partir du site de démarrage de la transcription (+1), en tenant compte de la région 5' UTR (région non codante transcrite avant le codon de démarrage), il serait raisonnable de supposer la présence d'une séquence régulatrice dans cette zone. Un programme spécifique a donc été créé à cet effet [14]. Celui-ci accepte au maximum **une mutation par boîte**.

2.4 Résultats

En examinant cette région, le programme a détecté une boîte consensus potentiellement importante pour la régulation. Toutefois, cette boîte présente deux mutations dans sa séquence : `[TTGACG]` pour la boîte ***-35*** et `[TATAAG]` pour la boîte ***-10***. Ces mutations, notamment une transition pour la boîte *-35* et une transversion dans la boîte *-10*, suggèrent que cette région pourrait ne pas correspondre à un promoteur fort. Ces résultats pourraient indiquer que la forte présence de la protéine codée par ce gène nécessite d'autres formes de régulation, qu'elles soient au niveau transcriptionnel ou traductionnel.

```
================================================================================
   Recherche de la présence de boîtes consensus en amont du gène
================================================================================

Boîte -35 trouvée à la position 3639948 : TTGACG
Boîte -10 trouvée à la position 3639969 : TATAAG
Nombre de séquences consensus trouvées : 1
Nombre total de séquences consensus trouvées : 1
```

FIGURE 6 – Capture d'écran des résultats du programme montrant la boîte consensus et ses mutations.

2.5 Analyse de la région en amont de *uspA*

Afin d'essayer d'identifier une éventuelle séquence régulatrice, j'ai souhaité orienter l'analyse vers la région en amont du gène. J'ai donc sélectionné une séquence **"query"** de **1000 pb** en amont de la CDS du gène *uspA*. L'idée principale était d'identifier une ou des séquences complexes sur-représentées dans le génome d'*Escherichia coli* K12 et également présentes en amont du gène *uspA*.

Pour ce faire, dans un premier temps, il fallait disposer d'un génome dit "contrôle" pour pouvoir comparer la fréquence d'apparition des séquences testées. J'ai donc commencé par analyser le génome complet d'*Escherichia coli* K12 à l'aide d'un petit programme, afin de comprendre sa composition en nucléotides : **A : 24.6%**, **T : 24.4%**, **C : 25.5%**, **G : 25.5%**. Ensuite, j'ai codé un programme [13] permettant de générer ce génome aléatoire à partir des données de composition en bases du vrai génome.

L'idéal aurait été de tenir compte de la complexité réelle d'un génome, mais j'ai voulu ne pas inclure certains aspects de la complexité biologique dans le modèle aléatoire, tels que les motifs exclus, afin de simplifier l'analyse. De plus, j'ai délibérément choisi de ne pas inclure la partie traductionnelle du gène, notamment l'analyse du *ribosome binding site* (RBS), ou encore la région en aval du gène, par souci de simplicité.

3 Algorithme de détection de motifs potentiellement régulateurs

Comme dit précédemment, l'objectif de cet algorithme [15] est d'analyser une séquence de 1 000 bases en amont du gène d'intérêt pour identifier des motifs qui pourraient jouer un rôle dans la régulation transcriptionnelle. L'algorithme procède en plusieurs étapes clés pour détecter des séquences répétées et les comparer à un génome aléatoire. Il est en partie inspiré de l'algorithme de Smith et Waterman et suit des heuristiques pour limiter son coût en ressources.

3.1 Étapes de l'algorithme

1. Découpage de la séquence en k-uplets La première étape consiste à découper la séquence en amont de 1 000 bases en sous-séquences de longueur fixe, appelées **k-uplets** (par exemple, des fragments de 6 à 10 bases). Chaque k-uplet est ensuite stocké pour un traitement ultérieur.

2. Recherche de motifs répétés Pour chaque k-uplet, le programme vérifie s'il apparaît plus de n fois (paramétrable dans le code source) dans le génome complet (séquence *query* exclue).

3. Extension du k-uplet Une fois qu'un k-uplet est identifié comme étant présent au moins n fois, il procède à une **extension** du k-uplet vers la gauche et vers la droite. Cela signifie qu'il continue à ajouter des bases adjacentes à ce k-uplet tout en vérifiant que la séquence étendue est toujours présente plus de n fois dans le génome complet.

Ce processus se poursuit tant que la séquence étendue est répétée au moins n fois. Si, à un moment donné, la séquence étendue n'est plus répétée (si un *gap* ou une divergence est rencontré), l'extension s'arrête, et un processus de test du k-uplet étendu est mis en place.

4. Vérification dans le génome artificiel L'algorithme compte alors combien de fois ce motif est présent dans le génome artificiel. Cette étape permet d'estimer la fréquence attendue de ce motif dans le génome réel, en se basant sur l'hypothèse nulle ($H0$) : la configuration observée des bases est attribuable au hasard.

5. Détection des séquences sur-représentées Si une séquence est **sur-représentée** dans le génome réel par rapport à sa fréquence attendue dans le génome aléatoire et que cette séquence est également présente dans la séquence *query* en amont du gène, le programme l'identifie comme un motif potentiel pouvant avoir un intérêt biologique.

Le **fold change**, calculé selon la formule suivante, est utilisé pour mesurer cette sur-représentation :

$$\text{Fold change} = \frac{\text{Occurrences dans la région réelle}}{\text{Occurrences attendues dans la séquence aléatoire}}$$

6. Limites du programme Il est important de noter que le programme défavorise fortement les séquences avec des *gaps*, puisque pour un *mismatch* d'une base, le *hit* potentiel n'est pas retenu s'il ne satisfait pas les paramètres entrés dans le code source (longueur minimale, *fold change* minimal, etc.).

Inversement, il favorise fortement les longues séquences (il est plus improbable qu'une longue séquence soit retrouvée dans un génome aléatoire). Cela tend donc à faire ressortir des séquences plus complexes qui réapparaissent à plusieurs reprises dans le génome.

Cela n'est pas toujours pertinent, car des séquences courtes, entrecoupées de paires de bases aléatoires, sont parfois impliquées dans une régulation génomique (par exemple, des boîtes consensus).

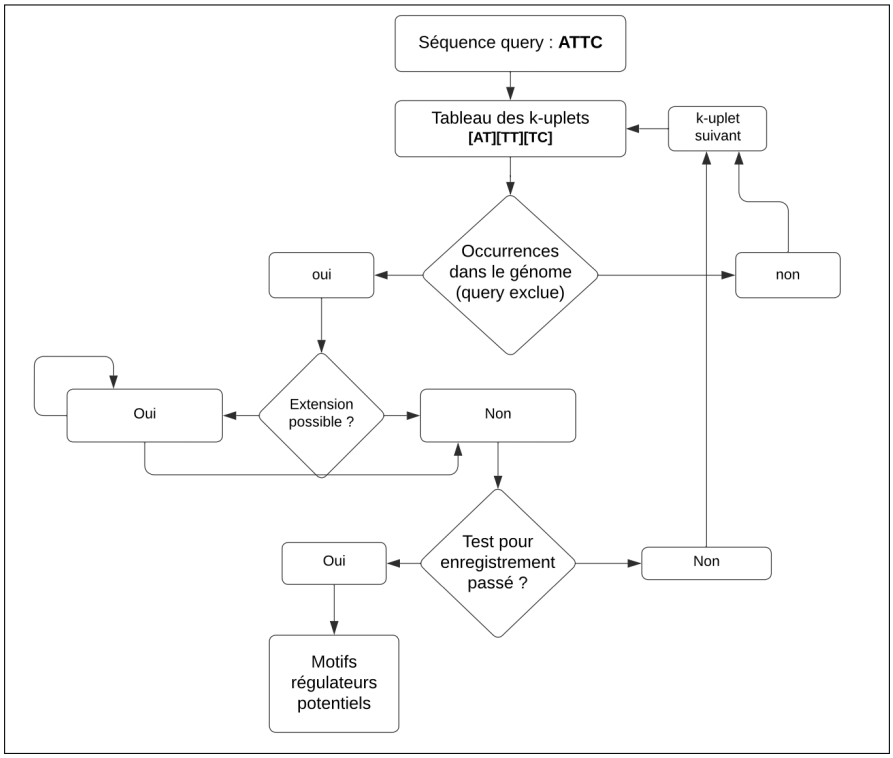

FIGURE 7 – Diagramme de flux de l'algorithme de détection des motifs régulateurs.

3.2 Résultats

Avec les paramètres suivants :

— `FOLD_CHANGE_MIN` : 1.5

— `LONGUEUR_K_UPLET` : 6

— `LONGUEUR_MIN_MOTIF` : 18

— `X_HIT_AVANT_ENTRE` : 20

— `LONGUEUR_REGION_ETUDIEE_MOTIFS` : 1 000

et le génome d'*Escherichia coli* K12 [1], un motif particulièrement intéressant a été identifié et est présenté ci-dessous.

```
------------------------------
Motif : GTAGGCCTGATAAGCGTAGCGCATCAGGC
Position : 3639277
Longueur : 29
Fold Change : 22000000.00
Occurrences réelles : 22
Occurrences aléatoires : 0
------------------------------
```

FIGURE 8 – Caractéristiques du motif identifié dans la région promotrice du gène *uspA*.

3.3 Interprétation

La **surreprésentation** de ce motif dans le génome et sa **présence dans la région promotrice** du gène *uspA*, combinée à sa **rareté dans la séquence aléatoire**, suggère que sa présence en amont du gène n'est pas le fruit du hasard.

3.4 Validation par criblage de bases de données

À ce stade de l'analyse bioinformatique, j'ai cherché à confirmer ou infirmer l'hypothèse selon laquelle l'algorithme avait mis en évidence un motif biologique pertinent. Pour ce faire, j'ai effectué un criblage de bases de données en utilisant la séquence découverte. J'ai utilisé l'outil **BLASTn** [8] afin de rechercher des alignements significatifs avec des séquences connues, présentes dans des bases de données publiques.

3.5 Résultats

Le criblage a révélé un alignement parfait avec *E. coli*, ainsi qu'avec d'autres organismes comme ***Shigella flexneri*** [10] et ***Citrobacter freundii*** [11]. Les résultats sont présentés ci-dessous et sont accessibles dans les références [9].

Organism	Blast Name	Score	Number of Hits	Description
Enterobacteriaceae	enterobacteria		100	
. Escherichia	enterobacteria		95	
. . Escherichia coli	enterobacteria	58.0	89	Escherichia coli hits
. . Escherichia coli O25b:H4-ST131	enterobacteria	58.0	2	Escherichia coli O25b:H4-ST131 hits
. . Escherichia albertii	enterobacteria	58.0	2	Escherichia albertii hits
. . Escherichia coli O157:H7	enterobacteria	58.0	1	Escherichia coli O157:H7 hits
. . Escherichia fergusonii	enterobacteria	58.0	1	Escherichia fergusonii hits
. Shigella flexneri	enterobacteria	58.0	2	Shigella flexneri hits
. Citrobacter freundii	enterobacteria	58.0	2	Citrobacter freundii hits
. Shigella flexneri 1b	enterobacteria	58.0	1	Shigella flexneri 1b hits

FIGURE 9 – Capture d'écran des résultats du BLASTn

3.6 Interprétation

Les alignements significatifs avec des séquences provenant d'autre organismes pourait suggéré que le motif identifié est c**onservé au cours de l'évolution** et pourrait jouer un rôle fonctionnel important chez les **Enterobacteriaceae**. L'identité élevée observée (**100 %**) renforce cette hypothèse. La **E-value inférieure à 0,05** prouve que l'alignement avec un autre organisme bactérien n'est pas le fruit du hasard.

Cependant, il est important de noter que la présence de notre motif chez une autre espèce de bactérie pourrait résulter d'un transfert horizontal entre les génomes de ces organismes, ce qui remettrait en question l'hypothèse d'une conservation évolutive.

4 Conclusion générale

Pour finir, la forte expression du gène *uspA* ne semble pas être due à un promoteur classique de type *-35/-10*, ni à une duplication du gène dans le génome. L'identification d'un motif spécifique dans la région promotrice, ainsi que les résultats du criblage de bases de données, peuvent suggérer une éventuelle implication du motif dans un ou plusieurs mécanismes biologiques spécifiques aux organismes appartenant à la famille des *Enterobacteriaceae*, bien que cette analyse puisse être nuancée.

En revanche, il n'est pas possible de conclure à un lien potentiel entre la séquence que nous avons mise en évidence et une implication directe dans la régulation transcriptionnelle. La distance entre le gène *uspA* et cette séquence est relativement élevée, ce qui complique l'établissement d'une relation entre les deux.

Il pourrait être intéressant d'étudier la séquence mise en évidence grâce à la génétique inverse (mutagénèse dirigée et étude du phénotype après mutation). Également, il pourrait être pertinent de vérifier si les 22 occurrences de la séquence identifiée (figure 8) se trouvent en amont de gènes, afin de valider cette hypothèse.

5 Remarques finales

Ce projet, bien qu'ayant ses limites et n'ayant pas vocation à être pris comme un travail de recherche, m'a permis d'acquérir une expérience pratique en bioinformatique, notamment en manipulant des séquences de données réelles, en programmant en C, et en réalisant des analyses bioinformatiques de base.

Je reconnais que certaines simplifications ont été faites, et j'espère avoir commis le moins d'erreurs possible. Également, le programme principal devra être amélioré dans une prochaine version (supprimer les doublons des "motifs", rendre la séquence *query* plus modulable, etc.). J'ai aussi eu recours à certains moments à des outils d'intelligence artificielle, notamment pour m'aider dans l'implémentation du code source.

Pour conclure, la réalisation de ce projet fut une expérience très enrichissante, et je tiens à vous remercier si vous avez pris le temps de vous y intéresser.

Références

[1] **Escherichia coli str. K-12 substr. MG1655 NCBI** (*NC_000913.3*). Disponible à l'adresse : https://www.ncbi.nlm.nih.gov/nuccore/NC_000913.3

[2] **PaxDb : Protein Abundance Database** Disponible à l'adresse : https://pax-db.org/

[3] **UspA universal stress protein A** (GenBank : 948007). Disponible à l'adresse : https://www.ncbi.nlm.nih.gov/gene/948007

[4] **Image de *Escherichia coli*** — Wikimedia Commons. Disponible à l'adresse : https://commons.wikimedia.org/wiki/File:EscherichiaColi_NIAID.jpg

[5] Pereira, R., Saraiva, J., & Cunha, J. (2017). *Energy efficiency across programming languages : how do energy, time, and memory relate ?* Dans : Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (pp. 256–267).

[6] **Structure 3D de la protéine UspA** (*1JMV*). Disponible à l'adresse : https://www.rcsb.org/structure/1JMV

[7] **UniProtKB - P0AED0** (*UspA* de *Escherichia coli* K12). Disponible à l'adresse : https://www.uniprot.org/uniprot/P0AED0

[8] **Basic local alignment search tool.** Disponible à l'adresse : https://blast.ncbi.nlm.nih.gov/Blast.cgi

[9] **Résultats du BLASTn**. Disponible à l'adresse : https://github.com/Biozinga/Ecoli-uspA-project/tree/version-1.0/Resultat_alignement_blastn

[10] **Shigella flexneri strain STLE1 chromosome, complete genome** (*GenBank : CP058826.1*). Disponible à l'adresse : https://www.ncbi.nlm.nih.gov/nuccore/CP058826.1

[11] **Citrobacter freundii isolate 112 genome assembly, chromosome : main** (*GenBank : OW848788.1*). Disponible à l'adresse : https://www.ncbi.nlm.nih.gov/nuccore/OW848788.1

[12] **Code source du programme de recherche du gène uspA** : Disponible à l'adresse : https://github.com/Biozinga/Ecoli-uspA-project/blob/version-1.0/src/recherche_gene.c

[13] **Code source du programme de génération d'une séquence aléatoire réaliste** : Disponible à l'adresse : https://github.com/Biozinga/Ecoli-uspA-project/blob/version-1.0/src/sequence_aleatoire.c

[14] **Code source du programme de recherche de boîte consensus** : Disponible à l'adresse : https://github.com/Biozinga/Ecoli-uspA-project/blob/version-1.0/src/recherche_consensus_box.c

[15] **Code source du programme de recherche de motifs complexes dans la région promotrice** : Disponible à l'adresse : https://github.com/Biozinga/Ecoli-uspA-project/blob/version-1.0/src/recherche_motifs.c