Devoir : Recherche et traduction de régions codantes dans le génome

UE Informatique, M1 Mathématiques et applications,
EADS, Université d'Aix-Marseille *

2024–2025

Ce devoir met en œuvre la recherche de sous-mots dans un contexte d'application à la génomique. Il est inspiré par l'article *À la recherche de régions codantes*[1] de François Rechenmann dans la revue en ligne *Interstices*, dont la lecture est par ailleurs chaudement recommandée.

1 Instructions

Ce devoir *fait partie de votre formation* : aller au bout devrait vous permettre à la fois de consolider certaines notions abordées dans les cours précédents, et d'exercer vos compétences en programmation.

L'objectif du devoir est la conception d'un programme, ou d'une bibliothèque de programmes, permettant de réaliser des opérations sur des données issues de la génomique. Les indications sont données en supposant que vous utilisez Python, mais vous pouvez utiliser un langage de programmation de votre choix, sous réserve de validation préalable.

La section 3, fixe des tâches que votre programme doit permettre de réaliser. Pour chacune de ces tâches, vous devez :

— produire le bout de programme correspondant,

— expliquer comment utiliser votre programme pour résoudre les objectifs annoncés,

— compléter ces informations en mentionnant les choix techniques que vous avez faits et, éventuellement, les difficultés que vous avez rencontrées (précisez, le cas échéant, ce qui ne fonctionne pas, ou en tout cas pas comme vous le souhaitez).

Ces explications et compléments peuvent être fournis dans les commentaires ou la documentation intégrée de votre programme, ou dans un fichier joint (soit un simple fichier texte type `README.md`, soit un document PDF).

Vous êtes libres de choisir votre style de programmation (utiliser des fonctions, des classes avec ou sans héritage, des modules, …) tant que votre code est correctement structuré et que les modalités d'utilisation sont claires. Le sujet ne prescrit pas d'organisation de votre code, ni de structures de données pour la représentation des différents objets considérés : c'est à vous de faire ces choix, et de rendre explicite quelle partie de votre code répond à chaque tâche.

1. `https://interstices.info/a-la-recherche-de-regions-codantes/`

1.1 Évaluation

Vous vous assurez d'obtenir la moyenne en produisant une tentative de code (même si elle plante dans certains cas ou ne réalise pas correctement tous les objectifs) pour les tâches des sections 3.1 à 3.3, sans erreur grossière ou flagrante (code refusé par l'interpréteur ou le compilateur), avec une interface clairement définie (par exemple, quelle fonction appeler, ou quelle classe instancier, et avec quels paramètres pour la tâche à réaliser), et en indiquant ce qui fonctionne et ce qui ne fonctionne pas, ou que vous n'avez pas réussi à traiter.

Si de plus votre code fonctionne sans erreur (sans soulever d'exception, sans générer de boucle infinie) sur des données typiques et produit des résultats de la nature attendue (même s'ils ne sont pas toujours corrects ou complets), vous obtenez au moins 12.

Si de plus votre code produit des résultats globalement corrects et complets (au moins pour tous les exemples mentionnés dans le sujet) vous obtenez au moins 14.

Sans rechercher l'optimisation à tout prix, on prêtera une certaine attention à l'efficacité des représentations choisies pour les données qu'on manipule : on aura à traiter des quantités importantes de données (le génome d'une simple bactérie peut comporter des millions de nucléotides), une certaine forme d'efficacité est donc nécessaire. Si en plus des autres conditions, votre code respecte cette attente, vous obtenez au moins 16.

Avec un programme correct, efficace, bien structuré, qui répond à tous les objectifs, avec un comportement prédictible y compris pour des données mal formatées, entièrement documenté, et avec une justification de vos principaux choix techniques : vous obtenez la note maximale.

2 Contexte

2.1 Le code génétique

On reprend ici les deux premiers paragraphes de l'article de François Rechenmann.

> *Stricto sensu*, le génome d'un organisme est l'ensemble de ses gènes ; autrement dit, l'information nécessaire à ses cellules pour synthétiser les protéines qui assurent des fonctions diverses : structure, transport, catalyse, *etc.* Par extension, le terme génome désigne également le support physique de cette information, la molécule d'ADN (Acide DésoxyriboNucléique), composant des chromosomes présents au sein de chacune des cellules de l'organisme. L'ADN est un enchaînement de nucléotides de quatre types différents distingués par leur base azotée : adénine, thymine, cytosine et guanine, et notés par les initiales A, T, C et G. Et c'est cet enchaînement qui code l'information génétique, au même titre qu'une suite de 0 et de 1 peut coder un son, une image ou une suite d'instructions.
>
> Au sein d'un gène, et plus précisément au sein de sa région codante (ou CDS pour CoDing Sequence), la suite des triplets de nucléotides, appelés codons, dicte la séquence en acides aminés de la protéine. La correspondance entre les 64 (4^3) codons possibles et les 20 acides aminés constitue le code génétique, identique à peu de variantes près chez tous les organismes vivants.

Résumons et précisons :

— le *génome* d'un organisme est une suite de *nucléotides* notés chacun par une lettre de l'ensemble $\mathcal{N} = \{\texttt{A}, \texttt{T}, \texttt{C}, \texttt{G}\}$, c'est-à-dire que le génome est un mot dans $\mathcal{N}^*$;

— un *codon* est une suite de trois nucléotides, c'est-à-dire un mot dans $\mathcal{C} = \mathcal{N}^3$;

— un *gène* est une suite de codons consécutifs, c'est-à-dire un mot dans $\mathcal{C}^*$;

Acide aminé	Lettre	Codons
Alanine	A	GCT, GCC, GCA, GCG
Arginine	R	CGT, CGC, CGA, CGG, AGA, AGG
Asparagine	N	AAT, AAC
Acide aspartique	D	GAT, GAC
Cystéine	C	TGT, TGC
Glutamine	Q	CAA, CAG
Acide glutamique	E	GAA, GAG
Glycine	G	GGT, GGC, GGA, GGG
Histidine	H	CAT, CAC
Isoleucine	I	ATT, ATC, ATA
Leucine	L	TTA, TTG, CTT, CTC, CTA, CTG
Lysine	K	AAA, AAG
Méthionine	M	ATG
Phénylalanine	F	TTT, TTC
Proline	P	CCT, CCC, CCA, CCG
Sérine	S	TCT, TCC, TCA, TCG, AGT, AGC
Thréonine	T	ACT, ACC, ACA, ACG
Tryptophane	W	TGG
Tyrosine	Y	TAT, TAC
Valine	V	GTT, GTC, GTA, GTG
stop		TAG, TAA, TGA

TABLEAU 1 – Code génétique : liste des codons codant chaque acide aminé.

— un gène apparait dans le génome, comme une *région codante*, c'est-à-dire un sous-mot du génome (dans $\mathcal{N}^*$), nécessairement de longueur multiple de 3, obtenu par concaténation des codons du gène ;

— chacun des 20 *acides aminés* est également noté par une lettre de l'alphabet, prise dans l'ensemble $\mathcal{A} = \{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{K}, \mathtt{L}, \mathtt{M}, \mathtt{N}, \mathtt{P}, \mathtt{Q}, \mathtt{R}, \mathtt{S}, \mathtt{T}, \mathtt{V}, \mathtt{W}, \mathtt{Y}\}$;

— une protéine est caractérisée par sa séquence d'acides aminés, c'est-à-dire un mot dans $\mathcal{A}^*$.

L'expression d'un gène consiste en la synthèse d'une protéine, dont la séquence d'acides aminés est obtenue par traduction des codons du gène. Chaque acide aminé est produit par un ou plusieurs codons, mais un même codon ne peut se traduire qu'en un acide aminé, au plus. Par ailleurs trois codons (`TAG`, `TAA`, `TGA`) ne se traduisent pas en acides aminés : on les appelle codons *stop*.[2] Formellement, le *code génétique* est donc une fonction partielle surjective des codons vers les acides aminés. Il est résumé dans le tableau 1, qui donne pour chaque acide aminé la liste des codons qui le produisent.[3] La dernière ligne est à part : elle liste les codons *stop*.

2.2 Régions codantes

On a décrit comment traduire la séquence de codons d'un gène pour obtenir la séquence d'acides aminés de la protéine correspondante. Mais comment identifier les *régions codantes* au

2. C'est en réalité un peu plus compliqué : dans certains contextes particuliers, un même codon peut se traduire en un acide aminé non standard, au lieu de sa traduction habituelle ; et l'un des codons *stop* peut produire un acide aminé dans certains cas. On néglige ces subtilités.

3. Le tableau est une version simplifiée du *tableau inverse* trouvé sur `https://fr.wikipedia.org/wiki/Code_génétique#Table_des_codons_d'ARN_messager`[W].

cadre de lecture	séquence de codons
$(1, 0)$	(GAG, TTA, TGC, AGT, GGT, AGT, ATG)
$(1, 1)$	(AGT, TAT, GCA, GTG, GTA, GTA, TGA)
$(1, 2)$	(GTT, ATG, CAG, TGG, TAG, TAT)
$(-1, 0)$	(AGT, ATG, ATG, GTG, ACG, TAT, TGA)
$(-1, 1)$	(GTA, TGA, TGG, TGA, CGT, ATT, GAG)
$(-1, 2)$	(TAT, GAT, GGT, GAC, GTA, TTG)

TABLEAU 2 – Les 6 séquences de codons pour le fragment d'ADN GAGTTATGCAGTGGTAGTATGA

sein d'un génome, c'est-à-dire les sous-mots du génome qui correspondent à des gènes ? Citons encore :

> Les biologistes savent que le début d'une région codante dans un génome bactérien est marqué par un triplet ATG, appelé *start*, et sa fin par l'un des triplets TAA, TAG ou TGA, appelés *stop*.

Trouver une région codante potentielle, c'est donc trouver un sous-mot du génome, de longueur multiple de trois, qui démarre par un codon *start* (inclus) et se termine par un codon *stop* (non inclus). Il reste quelques subtilités :

— on ne sait pas par avance dans quel sens il faut lire le génome pour obtenir une région codante (c'est-à-dire qu'il faut aussi considérer le mot obtenu en renversant le génome) ;

— les régions codantes ne sont pas nécessairement alignées sur des positions multiples de trois nucléotides.

On a donc deux *sens de lecture* (direct, disons 1, et inverse, disons -1), et trois *décalages* (0, 1 ou 2) possibles. Un *cadre de lecture* est la donnée d'un sens de lecture et d'un décalage, et on a donc 6 cadres de lecture possibles (les éléments de $\{-1, 1\} \times \{0, 1, 2\}$). Chaque cadre de lecture pour un génome induit une *séquence de codons* : un mot dans $\mathcal{C}^*$. Par exemple pour le fragment de génome GAGTTATGCAGTGGTAGTATGA, contenant 22 nucléotides, on obtient les 6 séquences du tableau 2. Comme $22 = 3 \times 7 + 1$, on obtient des séquences de 7 codons pour les décalages 0 et 1, et seulement 6 codons pour un décalage de 2.

Par ailleurs, le codon *start* ATG est lui-même traduit en un acide aminé : la méthionine (M). Et il peut figurer à l'intérieur d'une région codante, en plus du début. Donc si on peut deviner où s'arrêter (les codons *stop* ne sont jamais traduits en acides aminés), il n'est pas évident de savoir où on commence. À partir de ces informations, on ne peut donc qu'essayer de deviner où se trouvent les séquences potentiellement codantes, au risque d'obtenir quantité de faux positifs.

On applique deux heuristiques. D'abord on ne recherchera que les régions codantes raisonnablement grandes (typiquement, au moins 100 codons). Par ailleurs, on ne s'intéressera qu'aux régions codantes maximales : un sous-mot qui précède immédiatement un codon *stop*, qui ne contient pas de codon *stop*, et qui commence avec le codon *start* le plus à gauche possible.

Dans les séquences de codons du tableau 2, on trouve :

— une unique région codante (qui est donc maximale) en position 1 dans le cadre $(1, 2)$: (ATG, CAG, TGG), terminée par le codon *stop* TAG en position 4 ;

— deux régions codantes pour le cadre $(-1, 0)$: (ATG, ATG, GTG, ACG, TAT) en position 1 et (ATG, GTG, ACG, TAT), en position 2, toutes deux terminées par le codon *stop* TGA en position 6, seule la première étant maximale.

2.3 Bases de données publiques

On souhaite travailler sur des données réalistes, issues du monde réel.

On trouve des bases de données en ligne qui fournissent à la fois des génomes d'organismes séquencés, et les chaînes d'acides aminés correspondant à des protéines connues. Par exemple la *European Nucleotide Archive*[4] fournit des données de séquençage de génomes, et *UniProt*[5] fournit des données de séquençage de protéines.

On peut par exemple télécharger le génome d'une bactérie commune, *Bacillus subtilis*, depuis la page `https://www.ebi.ac.uk/ena/browser/view/GCA_046529535.1`, en cliquant sur le lien *All Seq FASTA*, ce qui devrait télécharger un fichier nommé `GCA_046529535.1.fasta.gz`.

De même, on peut télécharger la base de toutes les protéines manuellement annotées du jeu *Swiss-Prot* depuis la page `https://www.uniprot.org/help/downloads`, en cliquant sur le lien *fasta* de la ligne *Reviewed (Swiss-Prot)*. Ceci devrait télécharger un (gros : 90 Mo environ) fichier nommé `uniprot_sprot.fasta.gz`.

Dans les deux cas, on obtient un fichier au format FASTA,[6] compressé au format *gzip*.[7] Pour le décompresser, tout gestionnaire d'archives devrait convenir, mais on préfèrera laisser le programme accéder directement à la version compréssée (on verra plus loin comment le faire avec Python). Pour disposer d'exemples décompressés de taille raisonnable, deux fichiers FASTA sont joints au sujet :

— `bsubtilis_fragment.fasta` contenant un fragment de 6000 nucléotides du génome de *Bacillus subtilis* ;

— `sprot_extrait.fasta` contenant une dizaine de protéines issues du jeu *Swiss-Prot*.

Vous pouvez utiliser ces entrées pour tester votre programme, avant de le faire sur un génome complet et l'intégralité de la base *Swiss-Prot*.

Le format FASTA est extrêmement simple. Chaque fichier FASTA est un fichier texte, codé en ASCII, comportant trois types de lignes :

— une ligne qui commence par le caractère `";"` est un commentaire et peut être ignorée ;

— une ligne qui commence par le caractère `">"` est une description de section (donnée par tous les caractères suivant le `">"`) ;

— les autres lignes sont constituées de lettres latines majuscules,[8] et représentent le contenu d'une section.

Le fichier comporte une ou plusieurs sections, les unes à la suites des autres. Chaque section est constituée par une ligne de description, suivie de lignes de contenu : le contenu de la section est obtenu en concaténant les lignes de contenu.

3 Tâches

On rappelle que, tout au long de la réalisation des tâches suivantes, vous devez documenter vos choix techniques, en particulier les représentations des divers éléments en jeu (codons, génomes complets, régions codantes, protéines, *etc.*).

4. `https://www.ebi.ac.uk/ena/browser/home`
5. `https://www.uniprot.org/`
6. Voir `https://fr.wikipedia.org/wiki/FASTA_(format_de_fichier)`[W].
7. Voir `https://fr.wikipedia.org/wiki/gzip`[W].
8. Plus `"*"` et `"-"`, qui ont des significations spéciales, qu'on ignore ici.

3.1 Traduction de codons

Codon par codon. On rappelle que le code génétique est donné par le tableau 1. On voudrait, à partir d'un codon caractérisé par une chaîne de trois lettres dans $\mathcal{N}$, obtenir sa traduction en acide aminé, codée comme une lettre dans $\mathcal{A}$. Par exemple, pour le codon `CAG`, on devrait obtenir l'acide aminé `Q`. Cette traduction doit être obtenue en temps constant, aussi efficace que possible.

Le cas des codons *stop* est particulier, puisqu'ils ne se traduisent pas en acides aminés : pensez à expliciter la manière dont vous le traitez.

Traduction de séquences. Étant donnée une séquence de codons on veut pouvoir obtenir la séquence d'acides aminés correspondante.

Par exemple, pour la séquence de codons (`ATG`, `ATG`, `GTG`, `ACG`, `TAT`) on devrait obtenir la chaîne d'acides aminés `MMVTY`.

3.2 Lecture d'un génome

Cadres de lecture. Étant donnés un cadre de lecture et un génome (caractérisé par sa suite de nucléotides, c'est-à-dire comme une chaîne de caractères pris dans $\mathcal{N}$) on veut pouvoir parcourir la séquence de codons correspondante.

Testez par exemple sur les six cadres de lecture du génome fictif du tableau 2.

Recherche de régions codantes maximales. Étant donnée une séquence de codons, on veut pouvoir parcourir ses régions codantes maximales.

De plus, lorsqu'on manipule des régions codantes, on veut pouvoir retrouver le cadre de lecture et la position du génome pour lesquels elles sont obtenues.

Par exemple, en considérant comme génome fictif la séquence du tableau 2, on devrait obtenir les deux régions codantes maximales mentionnées plus haut, avec les cadres de lecture et positions annoncées.

Il sera utile de fournir un moyen de limiter les résultats aux régions codantes d'une taille supérieure à un certain seuil (disons 100 codons par défaut, mais ce seuil doit pouvoir être paramétrable).

3.3 Lecture de fichiers FASTA

Fichiers FASTA simples. On veut pouvoir ouvrir un fichier FASTA et lire son contenu. Plus précisément, on veut, en fournissant le nom d'un fichier FASTA, obtenir un objet qui représente la liste de ses sections : pour chaque section, on veut obtenir sa description et son contenu.

On veut également pouvoir chercher les sections dont la description contient une sous-chaine (par exemple, obtenir la liste des sections dont la description contient `subtilis`).[9] De même on veut pouvoir chercher quelles sont les sections dont la séquence contient une sous-chaine donnée.

3.3.1. *En Python*  . Si vous travaillez en Python, quelques rappels pourront être utiles. On ouvre un fichier `fichier.ext` avec `open("fichier.ext")`, qui retourne un objet itérable : quand on itère sur un fichier, on obtient successivement toutes ses lignes. Ainsi :

```
f = open("fichier.ext")
lignes = list(f)
f.close()
```

9. Idéalement, on voudrait même pouvoir rechercher des expressions rationnelles : `https://docs.python.org/fr/3.10/library/re.html`. Mais c'est parfaitement optionnel.

ouvre le fichier `fichier.ext`, assigne la liste de toutes ses lignes à la variable `lignes`, puis referme le fichier. Une manière équivalente et plus *pythonesque* de faire la même chose :

```
with open("fichier.ext") as f:
    lignes = list(f)
```

le bloc `with` s'assurant que le fichier est bien fermé à la fin.

Notez que chaque ligne se termine par un caractère de fin de ligne explicite : vous pouvez vous en débarrasser avec la méthode `strip` des chaînes, ou bien en supprimant à la main le dernier caractère.

Par défaut, `open` ouvre les fichiers en mode texte, et les lignes sont des chaînes de caractères `str`. Suivant le choix de représentation que vous avez fait, vous préférerez peut-être obtenir des chaines d'octets `bytes` : pour cela, vous pouvez utiliser l'argument optionnel `mode`. Par exemple :

```
with open("fichier.ext", mode="rb") as f:
    lignes = list(f)
```

(`'rb'` signifie *read binary*, c'est-à-dire lecture en mode binaire ; `mode` vaut `'rt'` par défaut, c'est-à-dire lecture en mode texte).

Vérifiez que votre code fonctionne avec les fichiers exemples fournis : `bsubtilis_fragment.fasta` et `sprot_extrait.fasta`.

Fichiers FASTA compressés. On veut aussi pouvoir ouvrir directement les fichiers FASTA compressés en `gzip`.

3.3.2. *En Python* . Pour ça vous pourrez utiliser le module `gzip` de la bibliothèque standard de Python. Par exemple :

```
import gzip
with gzip.open("fichier.ext") as f:
    lignes = list(f)
```

Notez que `gzip.open` admet aussi l'argument optionnel `mode`, mais celui-ci vaut `'rb'` par défaut.

Idéalement, votre solution devrait utiliser `gzip.open` pour les fichiers dont le nom se termine en `.gz`, et `open` pour les autres.

Vérifiez que votre code fonctionne avec des fichiers téléchargés depuis les bases publiques : par exemple les fichiers `GCA_046529535.1.fasta.gz` et `uniprot_sprot.fasta.gz` mentionnés plus haut.

3.4 Applications

Dans ce qui suit, à chaque fois qu'on écrit *région codante*, on veut dire *région codante maximale de taille supérieure à un seuil donné.*

Une fois qu'on sait :

- charger un fichier FASTA contenant un génome ;
- rechercher les régions codantes dans ce génome ;
- traduire une région codante en séquence d'acides aminés ;
- charger un fichier FASTA contenant une base de données de protéines ;

on voudrait pouvoir :

- obtenir la proportion de codons dans un cadre de lecture qui sont potentiellement exprimés, c'est-à-dire qui font partie d'une région codante ;

— obtenir la liste des régions codantes qui codent pour des protéines connues, avec la description de la protéine pour chaque région codante ;

— obtenir la proportion de codons dans un cadre de lecture qui sont effectivement exprimés, c'est-à-dire qui font partie d'une région codante qui code pour une protéine connue.

Avant de vous confronter aux données effectivement issues des bases publiques, vous pouvez d'abord tester vos contributions avec les extraits fournis (`bsubtilis_fragment.fasta` et `sprot_extrait.fasta`) : il y a seulement 3 régions codantes maximales comptant plus de 100 codons dans cet extrait de génome, dont 2 en sens direct, et 1 en sens inverse, et seules les 2 en sens direct correspondent à des protéines.

Vous pouvez indiquer le résultat de vos tests dans votre rendu, en indiquant sur quels jeux de données vous les avez faits (merci toutefois de ne pas inclure directement les gros fichiers FASTA dans votre rendu).