Rapport de démarche scientifique

DÉTERMINATION DE L'ORIGINE DE RÉPLICATION

Table des matières

I. Introduction:

La **Bio-Informatique** est un domaine en plein progression qui relie la biologie et l'informatique pour déchiffrer les mystères de la vie à l'échelle moléculaire.

Dans ce rapport, nous explorerons comment cette discipline nous aide à comprendre les séquences d'ADN et de protéines et comment elle peut être appliquée à des problèmes biologiques concrets.

Au cœur de notre projet nous avons étudié **la bactérie Borrelia burgdorferi B31**, responsable de la maladie de Lyme chez l'homme. Comprendre les mécanismes de cette bactérie peut aider à développer des stratégies de diagnostic et de traitement plus efficaces.

Notre objectif est d'apprendre à utiliser des outils bio-informatiques pour une analyse spécifique de la séquence d'ADN de cette bactérie, et d'identifier son origine de réplication.

La maladie de Lyme est une maladie infectieuse complexe transmise par les tiques et causée par Borrelia burgdorferi. Pour mieux comprendre cette bactérie et ses mécanismes de réplication, nous nous concentrons sur son origine de réplication, un site spécifique sur son génome où commence la réplication de l'ADN.

Au cours de ce projet, notre premier objectif est de développer un programme Python capable de vérifier qu'une séquence donnée est une séquence d'ADN et de calculer sa taille et son taux de GC, une mesure importante pour comprendre la stabilité et la fonction de la séquence.
Ensuite, nous appliquerons ce programme pour analyser les chromosomes du poulet et créer un graphique illustrant la relation entre le pourcentage de GC et la taille des chromosomes.

En parallèle, nous utiliserons R pour générer des graphiques qui mettent en évidence les caractéristiques distinctives de la séquence d'ADN de Borrelia burgdorferi B31, en particulier son taux de GC et aussi son biais de GC.

En conclusion, notre projet combine l'apprentissage des langages de programmation Python et R à la résolution d'une problématique biologique importante. En identifiant l'origine de réplication de Borrelia burgdorferi B31, nous pouvons apporter des connaissances précieuses pour la prévention et le traitement de la maladie de Lyme.

II. Matériel et Méthode :

1. Avantages et inconvénient de Python:

Python est un langage de programmation interprété, facile à apprendre et à utiliser. Sa syntaxe simple, claire et lisible facilite la compréhension du code. De plus, la simplification de l'utilisation des fonctions, car il n'est pas nécessaire de déclarer explicitement le type des variables, des fonctions, ...

Une autre caractéristique appréciée de Python est la description de l'erreur et de la ligne précise de cette faute, ce qui facilite le repérage et la correction des erreurs.

Aussi, la compatibilité de Python avec différentes plateformes en fait un bon choix pour notre projet.

Cependant, malgré ses avantages, Python peut présenter quelques inconvénients. Tout d'abord, il est généralement plus lent que les langages de programmation compilée, ce qui peut poser des problèmes lors du traitement de grandes quantités de données ou lors de l'exécution de calculs intensifs.

Bien que sa performance puisse être inférieure à celle de certains autres langages dans ces situations, Python reste un choix très populaire en raison de sa simplicité, de sa polyvalence et de sa large communauté de développeurs.

2. Avantages et inconvénients de R :

Le langage R est largement utilisé dans le domaine des statistiques et de l'analyse de données ce qui est donc aussi le cas pour le domaine de bio-informatique. Il offre de nombreux avantages, notamment sa capacité à manipuler, analyser et visualiser efficacement des données.

Il dispose une large gamme de bibliothèques et de fonctions qui facilitent le traitement des données, en particulier dans le domaine de la recherche scientifique.
De plus, R offre une grande capacité dans la création de graphiques de haute qualité, permettant aux utilisateurs de représenter leurs données de manière claire et précise à l'aide d'une grande variété de visualisations.

Cependant, l'apprentissage de R peut être difficile pour les débutants en raison de sa syntaxe complexe et de ses concepts statistiques avancés. Certaines opérations peuvent être plus lentes dans R par rapport à d'autres langages de programmation, surtout lorsqu'il s'agit de manipuler de grandes quantités de données.

Malgré ses difficultés, R reste populaire pour l'analyse de données en raison de sa puissance et de sa capacité d'adaptation, et de créer des graphiques de haute qualité en fait un outil précieux pour comprendre et communiquer les résultats.

3. Séquences & Analyse :

Dans le cadre de notre étude, nous disposons du génome complet de **Borrelia Burgdoferi**, l'agent pathogène responsable de la maladie de Lyme. Ce génome, qui a une longueur totale de 910 724 paires de bases, a été mis à notre disposition via la plateforme eCampus, et qui est représenté dans un fichier au format FASTA.

Le format FASTA, largement utilisé en bio-informatique, permet de représenter de
manière claire et lisible des séquences biologiques telles que l'ADN, l'ARN ou les protéines. Chaque séquence est précédée par un en-tête débutant par le symbole ">", suivi immédiatement par des informations descriptives telles que le nom de la séquence, son origine et le nombre de nucléotides. Ce format permet non seulement le stockage et l'échange de données, mais aussi leur traitement informatique, grâce à sa structure claire et à sa flexibilité pour annoter les séquences.

Donc pour démarrer notre analyse, nous avons commencé par examiner une petite séquence nommée **seq_TD1**, comprenant 2330 nucléotides. Notre objectif initial était de développer un code Python robuste capable de vérifier que la séquence est une séquence d'ADN valide, de calculer sa taille et son taux de GC. (Fichier 1)

Par la suite, nous avons adapté notre code Python afin de traiter les fichiers correspondant aux chromosomes de poulet. Cette adaptation a permis de lire les données de chaque chromosome et d'écrire les résultats dans des fichiers de sortie spécifiques à chaque chromosome. (Fichier 2)
Cette étape était essentielle pour faciliter la lecture des données dans R et pour créer des graphiques permettant une visualisation plus claire et précise des résultats obtenus.

Nous avons également utilisé **la technique de la fenêtre glissante**. Cette méthode
implique le déplacement d'une fenêtre de taille fixe le long de la séquence, ce qui permet d'analyser des sections successives pour détecter des motifs ou des régions d'intérêt. L~~e but de cette approche~~ était d'obtenir une analyse fine des séquences génomiques afin d'identifier des zones potentiellement significatives, notamment pour déterminer les origines de réplication.

Finalement, nous avons réussi à développer un code Python complet pour étudier la séquence de Borrelia Burgdorferi (Fichier 3). Nous avons vérifié la séquence, calculé le taux de GC et le skew GC en intégrant la technique de la fenêtre glissante.

Cependant, nous avons rencontré des difficultés lors de la vérification de la validité de la séquence. Et après une vérification visuelle, nous avons remarqué la présence de 3 lignes vides. Par conséquent, notre programme a été modifié pour les supprimer.

Mais malgré cette modification, la séquence reste invalide. Nous avons alors essayé de trouver le problème, et avec deux méthodes, nous avons pu identifier la présence des lettres 'R', 'Y', 'K', 'M', 'S', 'W' et 'N', en plus des bases standards 'A', 'T', 'G' et 'C'.

➢ **Première méthode : (différence de taille)**

```
print(nb_TOTAL)       ----> 910683
print(len(sequence)) ----> 910724
```

Avec : *nb_TOTAL = nbT + nbG + nbA + nbC + nbN*

➢ **Deuxième méthode :**

```
def verifier_sequence_adn(sequence):
    nucleotides_valides = {'A', 'T', 'G', 'C','N'}
    for nucleotide in sequence:
        if nucleotide not in nucleotides_valides:
            print(f"Nucléotide invalide : {nucleotide}")
            return False
    return True
```

Résultats : *'R', 'Y', 'K', 'M', 'S', 'W'*

Nous avons ensuite recherché la signification de ces lettres et avons découvert que:

- 'N' : désigne une position dans la séquence où la base nucléique est inconnue ou non spécifiée.
- 'R' : indique une position où se trouve une purine (A ou G).
- 'Y' : signale une position où se trouve une pyrimidine (C ou T).
- 'K' : représente une position où se trouve une base nucléique cétonique (G ou T).
- 'M' : désigne une position où se trouve une base nucléique amino (A ou C).
- 'S' : indique une position où se trouve une base nucléique forte/faible (G ou C).

III. Résultats :

1. Chromosomes du Poulet

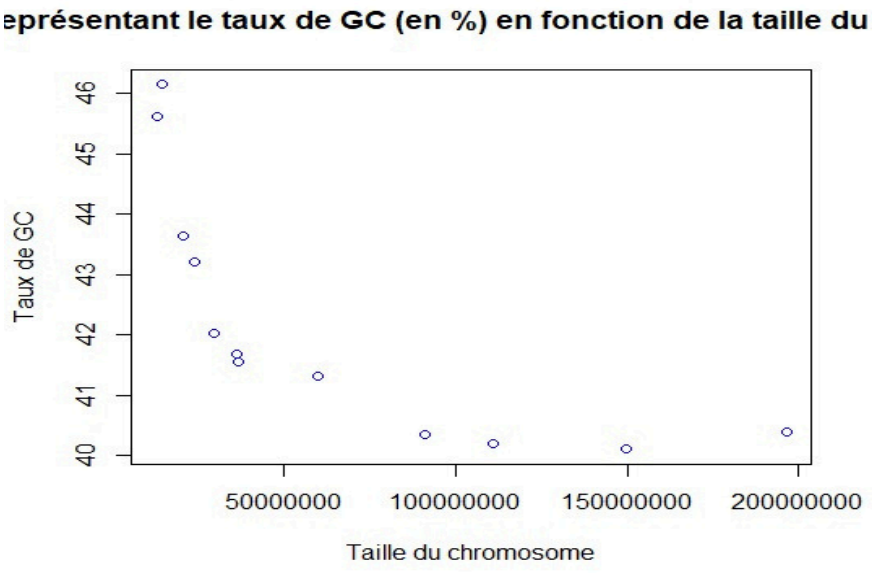
A. Tableau 1. Extrait des résultats

Voici le tableau des résultats obtenus dans le dossier sortie grâce à python :

Chromosomes	*Taille*	*Taux de GC (%)*
Gallus_chr1.txt	196449156	40.39
Gallus_chr2.txt	149539284	40.10
Gallus_chr3.txt	110642502	40.20
Gallus_chr4.txt	90861225	40.34
Gallus_chr5.txt	59506338	41.31
Gallus_chr6.txt	36220557	41.68
Gallus_chr7.txt	36382834	41.56
Gallus_chr8.txt	29578256	42.02
Gallus_chr9.txt	23733309	43.21
Gallus_chr10.txt	20453248	43.64
Gallus_chr15.txt	12703657	45.62
Gallus_chr20.txt	14265659	46.15

B. Graphe 1 : taux de GC (%) en fonction de la taille

Après implémentationdansR(Annexe1),onaobtenulegraphesuivant :



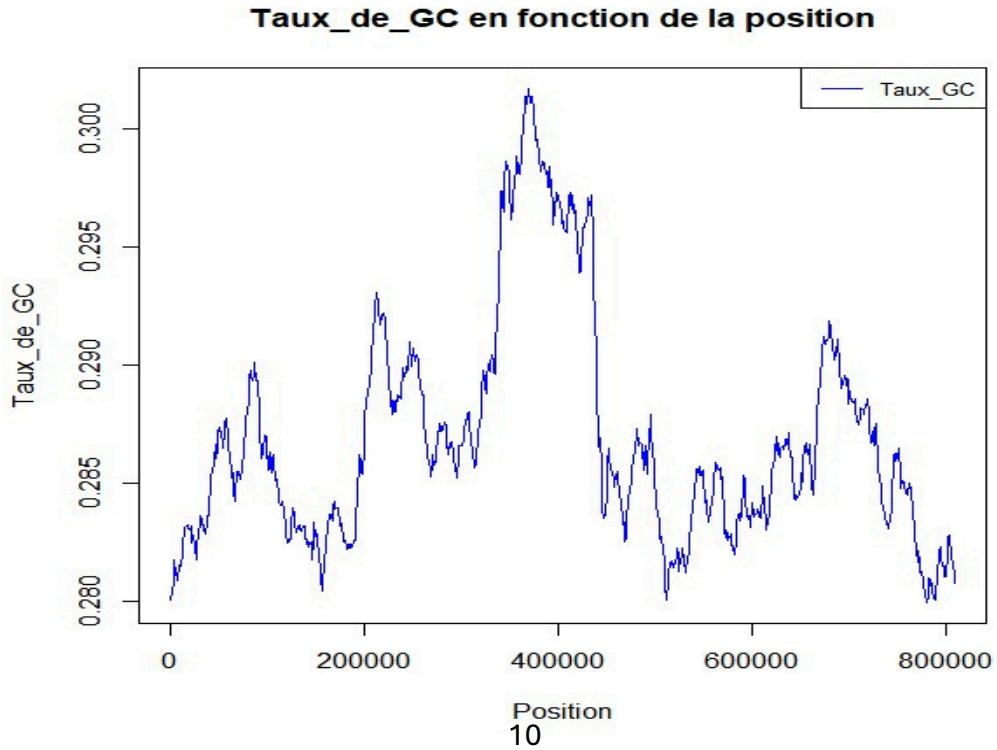
2. Borrelia burgdorferi B31 :

A. Tableau 2. Extrait des résultats

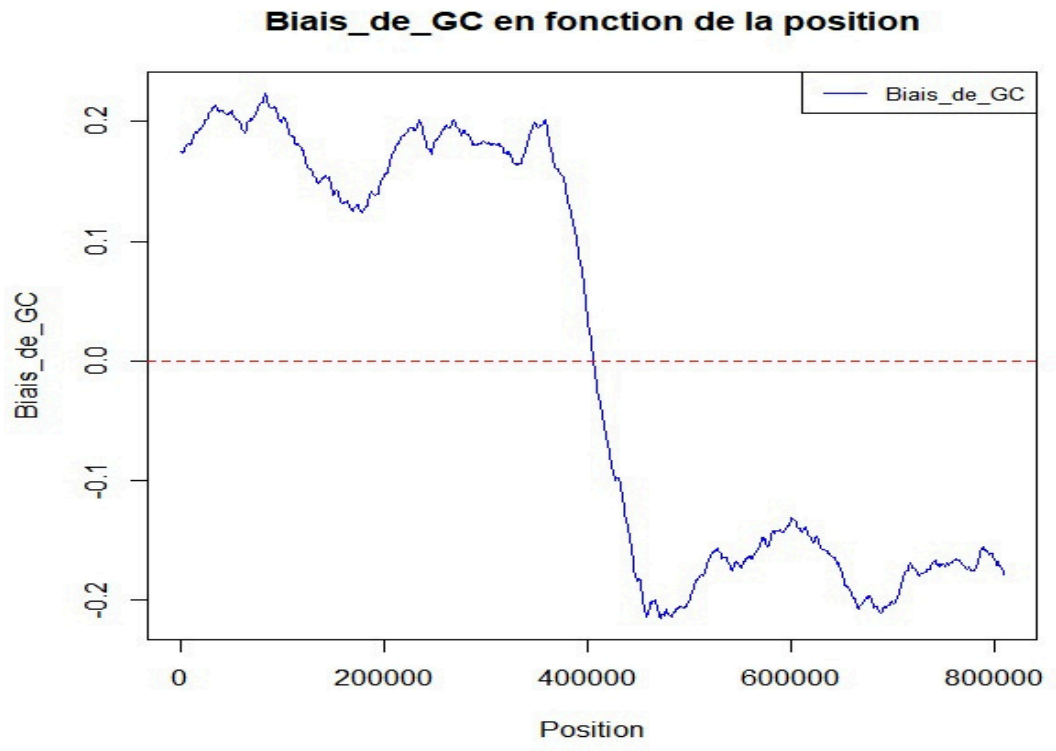
Voici un extrait des résultats obtenus dans le fichier sortie_taux_gc et le fichier sortie_skew_gc grâce à python :

Position	***Taux GC***	***Biais GC***
0	0.28007	0.17470632341914522
10000	0.28155	0.1811045995382703
50000	0.28737	0.20924244005985315
100000	0.28554	0.20319394830846815
200000	0.28778	0.15602196122037668
300000	0.28659	0.18266513137234378
400000	0.29717	0.03879934044486321
500000	0.28509	-0.20025255182573923
600000	0.28418	-0.1338588218734605
700000	0.28837	-0.20158130179977113
800000	0.28103	-0.16521367825499056

B. Graphe 2 : Taux GC en fonction de la position



C. Graphe 3 : Biais GC en fonction de la position



Vous pouvez trouver les codes R de ces deux graphes dans, (Annexe 2), (Annexe 3).

IV. Discussion et perspectives :

Les résultats de nos calculs du taux de GC et du skew GC pour la séquence d'ADN de Borrelia burgdorferi B31 offrent des perspectives intéressantes sur la structure et la fonction de son génome, ce que nous les a aussi confirmés les graphiques obtenus de ces derniers en fonction de la position dans la séquence (la fenêtre glissante).

1. Discussion des Résultats :

Taux de GC :

Le graphique du taux de GC en fonction de la position montre des variations significatives dans la composition en bases de l'ADN. On observe des fluctuations dans le taux de GC le long du génome. Certaines régions présentent des pics, indiquant une concentration plus élevée de paires de bases G-C, tandis que d'autres montrent des creux, suggérant une prédominance de paires de bases A-T.

Skew GC :

Le graphique du skew GC en fonction de la position révèle des variations dans la répartition relative des bases G et C le long de la séquence. Un skew positif indique une prédominance de la base C par rapport à la base G, et un skew négatif indique l'inverse. Une ligne de référence à y=0 est tracée pour faciliter la détection des changements de direction du skew, ce qui pourrait indiquer des régions potentielles d'origine de réplication.

2. Perspectives :

Origine de Réplication : Les régions où le skew GC passe de positif à négatif pourraient indiquer les sites potentiels d'origine de réplication (OriC). Les variations dans le skew peuvent être utilisées pour identifier ces sites cruciaux pour la réplication de l'ADN.

Structure du Génome : Les fluctuations du taux de GC peuvent refléter la structure et la fonction différentielle des régions du génome. Par exemple, les régions riches en GC pourraient correspondre à des gènes ou à des régions fonctionnelles importantes, tandis que les régions pauvres en GC pourraient être associées à des régions non codantes ou à des régions répétées.

Évolution et Adaptation : Les variations dans la composition de l'ADN peuvent nous aider à comprendre comment Borrelia burgdorferi B31 évolue et s'adapte à son environnement. Il serait intéressant d'explorer comment ces variations sont liées à des caractéristiques importantes telles que la capacité pathogène, la résistance aux antibiotiques ou d'autres aspects biologiques de la bactérie dans de futures études.

En résumé, l'analyse du taux de GC et du skew GC nous donne un aperçu important de la structure et du fonctionnement du génome de Borrelia burgdorferi B31. Ces résultats pourraient avoir des conséquences importantes en biologie moléculaire, en évolution et dans la compréhension des mécanismes de la maladie.

V. Annexes :

Fichier 1. *Code python – Seq_TD1 (Objectif 1)*

Fichier 2. *Code python – Chromosomes du poulet (Objectif 2)*

Fichier 3. *Code python – Borrelia burgdorferi B31*

Annexe 1. *Code R - taux de GC (%) en fonction de la taille*

```r
# Définir le répertoire de travail
setwd("H:/Demarches_Scientifique")

# Charger les données depuis le fichier
data <- read.table("Rpoulet_resultats.txt", sep = "\t", header = TRUE)

# Ouvrir une nouvelle fenêtre graphique
dev.new(width = 8, height = 6)

# Empêcher les valeurs de se mettre sous forme exponentielle
options(scipen = 999)

# Tracer le graphe
plot(data[, 3] ~ data[, 2], xlab = "Taille du chromosome", ylab = "Taux de GC",
     main = "Graphe représentant le taux de GC (en %) en fonction de la taille
     du chromosome")

# Ajouter des points représentant les données
points(data[, 3] ~ data[, 2], col = "blue")

# Afficher le résumé des données
summary(data)
```

Annexe 2. *Code R - taux de GC en fonction de la position*

```
#ajouter une option pour éviter d'avoir des valeurs affichées en exponentielles
options(scipen = 999)

#tracage du grapique
plot(data$Position, data$Taux_GC,
     type = "l",
     main = "Le_taux_de_GC en fonction de la position",
     xlab = "Position",
     ylab = "Taux_de_GC",
     col = "blue", # Couleur de la courbe
     xlim = c(min(data$Position), max(data$Position)), # Limite de l'axe x
     ylim = c(min(data$Taux_GC), max(data$Taux_GC))) # Limite de l'axe y

#afficher des axes
axis(side = 1, at = x_seq, las = 2)
axis(side = 2, at = y_seq, las = 2)

# Réinitialiser l'option pour la notation exponentielle par défaut
options(scipen = 0)

#la légende
legend("topright",
       legend = "Taux_de_GC",
       col = "blue",
       lty = 1,
       cex = 0.8)
```

Annexe 3. *Code R – biais GC en fonction de la position*

```r
#décrire le chemin du fichier sortie qui contient les résultat de notre code
fichier <- "C:/Users/ayaza/OneDrive/Documents/demarche/sortie_skew_gc.txt"
#lecture du fichier
data <- read.table(fichier, header = TRUE)

#ajouter une option pour éviter d'avoir des valeurs affichées en exponentielles
options(scipen = 999)

#tracage du grapique
plot(data$Position, data$Skew_GC,
     type = "l",
     main = "Skew_GC en fonction de la position",
     xlab = "Position",
     ylab = "Skew_GC",
     col = "blue", # Couleur de la courbe
     xlim = c(min(data$Position), max(data$Position)), # Limite de l'axe x
     ylim = c(min(data$Skew_GC), max(data$Skew_GC))) # Limite de l'axe y

#afficher une ligne de référence(y=0)
abline(h = 0, lty = 2, col = "red") # Ligne en pointillés rouges pour la référence

#afficher des axes
axis(side = 1, at = x_seq, las = 2)
axis(side = 2, at = y_seq, las = 2)

# Réinitialiser l'option pour la notation exponentielle par défaut
options(scipen = 0)
#la légende
legend("topright",
       legend = "Skew_GC",
       col = "blue",
       lty = 1,
       cex = 0.8)
```

FIN