université de BORDEAUX

ANNEE UNIVERSITAIRE 2020-2021
SESSION 1 D'AUTOMNE DECEMBRE 2020
MENTION : BIOINFORMATIQUE
Code UE : 4TBI703U
Intitulé de l'épreuve : Omiques et Bioinformatiques

Date : 15/12/2019 **Heure : 9h** **Durée :** 1h30
Documents : ~~autorisés~~ / non autorisés
Epreuve de M/Mme : P. Thébault, R. Uricaru et J.C. Taveau

Collège Sciences et technologies

Masters

1. Exercice Phylogénie (8 points)

La figure suivante (Fig.1) est issue de Davda S. and Martin B.R., MedChemComm, 2014. Elle porte sur une étude de l'évolution de la famille des gènes APT effectuée à partir de séquences d'invertébrés et vertébrés.

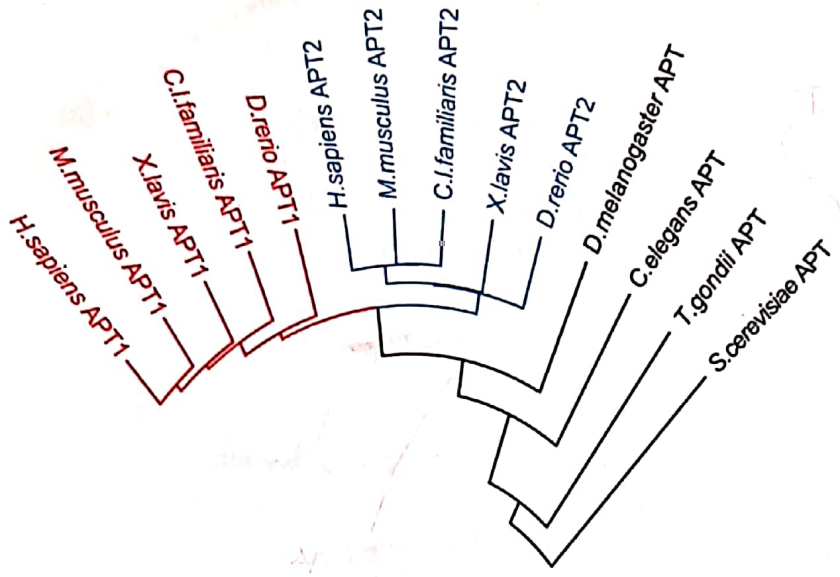

Fig.1*:* Shared homology between vertebrate and invertebrate acyl protein thioesterases.

Groupe des vertébrés: *Danio rerio, Mus musculus, Homo sapiens, Certhia familiaris, Xenopus laevis.* Groupe des invertébrés: *Saccharomyces cerevisiae, Toxoplasma gondii, Caenorhabditis elegans, Drosophila melanogaster.*

1. Décrivez les différentes étapes qui ont été réalisées pour construire cet arbre.
2. L'arbre n'a pas été raciné. Proposez deux méthodes pour raciner cet arbre.
3. Donnez au format NEWICK la partie de l'arbre se rapportant au gène APT1 concernant 5 TAXA.
4. A partir de la figure décrivez l'évolution de la famille des gènes APT chez les vertébrés et invertébrés.

2. Exercice Python (12 points)

Dans cet exercice nous allons manipuler une liste de *reads (lectures issues du séquençage d'un morceau d'ADN),* c'est à dire une liste de chaînes de caractères de même taille correspondant à des séquences nucléotidiques (composées de caractères 'A', 'C', 'G' et 'T'). Par exemple, la liste ***L_reads = ["ATTGC","AGTGC","ATTGA","ATTGC","CTTGG"]*** contient 5 reads de longueur 5.

Préambule

- Pour rappel, en langage Python :
  - tout comme on peut accéder à un élément d'une liste, on peut directement accéder à un caractère d'une chaîne **s** en utilisant la notation indicée **s[i]**, qui nous donne le (i+1)-ème caractère contenu dans la chaîne s ;

- la fonction **len** appliquée sur une séquence (liste, chaîne de caractères, ...) donne la longueur de celle ci, par exemple ***len(L)*** renvoie 6, ***len("ATTGC")*** renvoie 5.

- Dans les questions suivantes ***vous allez exclusivement vous servir de la notation indicée pour les chaînes de caractères***.
- Sauf mention contraire, à part la fonction ***len***. vous n'utiliserez aucune autre fonction mise à disposition dans le langage ***python***.
- Les ***valeurs passées en paramètre des fonctions seront considérées comme étant valides***, il n'y a pas besoin de faire des vérifications supplémentaires.

2.1. Écrivez une fonction ***nb_nt(seq, nt)*** qui calcule et renvoie le nombre d'occurrences du nucléotide ***nt*** (caractère 'A', 'C', 'G' ou 'T') dans une séquence nucléotidique ***seq*** (chaîne de caractères composée de 'A', 'C', 'G' et 'T'). Par exemple ***nb_nt("ATTGC", 'T')*** renverra 2.

2.2. En vous servant de la fonction précédente, écrivez une fonction ***moyenne_occ_nt(L, nt)*** qui calcule et renvoie la moyenne du nombre d'occurrences du nucléotide ***nt*** dans les reads contenus dans la liste de reads ***L***. Par exemple ***moyenne_occ_nt(L_reads, 'T')*** renverra (2+1+2+2+2+2)/6=1.833.

2.3. En utilisant la fonction précédente, écrivez une fonction ***moyenne_occ(L)*** qui pour une liste de reads ***L*** crée et renvoie une liste de nombres flottants contenant les moyennes de nombres d'occurrences dans les reads de la liste L pour chaque nucléotide dans l'ordre 'A', 'C', 'G', 'T'. Par exemple ***moyenne_occ(L_reads)*** renverra la liste [1, 0.66, 1.5, 1.833]. Dans cet exercice vous utiliserez la fonction ***append*** qui, appelée pour une liste ***myList*** et un élément ***e*** de la façon suivante ***myList.append(e)***, rajoute l'élément ***e*** à la fin de la liste ***myList***.

2.4. Écrivez une fonction ***nt_majoritaire(L, pos)*** qui prend en paramètre une liste de reads ***L*** et une position sur les reads (***0 <= pos < longueur(read)***) et qui renvoie le nucléotide majoritairement présent dans les reads de ***L*** sur la position ***pos***. Par exemple ***nt_majoritaire(L_reads, 4)*** renverra 'C', 'C' étant le nucléotide le plus présent sur la position 4 dans les reads de ***L_reads***. Dans cet exercice vous utiliserez la fonction ***max*** qui, appelée pour une liste ou un tuple de valeurs, renvoie la valeur maximale contenue dans la liste ou dans le tuple, par exemple ***max([1, 3, 2])*** *renvoie* ***3***, ***max(1, 3, 2)*** renvoie ***3*** également.

2.5. Écrivez une fonction ***logo(L)*** qui pour une liste de reads ***L***, construit et renvoie une chaîne de caractères de la même taille que les reads dans ***L***, appelée chaîne *logo*. La chaîne logo sera composée des nucléotides majoritaires pour chacune des positions. Par exemple ***logo(L_reads)*** renverra la chaîne ***"ATTGC"*** car dans ***L_reads*** A est le nucléotide majoritaire sur la position 0, T sur la position 1, etc. **Rappel** : L'opérateur + permet de concaténer deux chaînes de caractères.

2.6. Écrivez une fonction ***compare_seq(seq1, seq2)*** qui compare deux séquences de nucléotides ayant la même taille et renvoie le nombre de différences entre les deux séquences. Par exemple ***compare_seq("ATTGC", "AGTGC")*** renvoie 1 car les séquences diffèrent uniquement sur la position 1 ('T' versus 'G').

2.7. Vous avez peut être remarqué qu'utiliser la fonction ***moyenne_occ_nt*** dans la fonction ***moyenne_occ*** n'est pas le plus efficace. Écrivez une deuxième version de la fonction ***moyenne_occ(L)*** qui calcule la même chose que la version initiale mais en faisant un unique parcours de chaque read de la liste de reads ***L***.