le cnam

Projet Tuteuré

Logiciel de reconstruction d'arbre phylogénétique en Java

EAP Guillaume

2023

Superviseur académique/Enseignant référent : SPADONI Jean-Louis

Établissement/Formation : Cnam Paris – Licence professionnelle de biotechnologie, option bio-informatique

Date de la soutenance : Octobre 2023

Table des matières

Glossaire

UPGMA : Unweighted Pair Group Method/Méthode de Regroupement par Paires Non Pondérées avec Moyenne Arithmétique

NJ : Neighbor Joining

Dendrogramme : représentation d'un arbre avec une structure arborescente

FASTA : Format fichier de texte pour les séquence nucléiques et protéiques.

VIH : Virus de l'Immunodéficience Humaine

VIS : Virus de l'immunodéficience simienne

GUI : Graphical User Interface/Interface Graphique Utilisateur

Classe : En Java, une classe est un modèle ou un plan pour créer des objets.

Package : En Java, un "package" est un mécanisme d'organisation et de gestion des classes et des interfaces.

Nœud : Point clé qui montrent les relations évolutives entre les organismes ou les séquences étudiées dans un arbre phylogénétique.

Feuille : Nœud externe, qui à l'inverse du nœud interne, ne se subdivise pas. Elle représente une entité existante.

Cluster : Groupe de taxons, d'espèces ou de séquences qui partagent un ancêtre commun récent dans un arbre phylogénétique.

Matrice des distances : Dans le cas de la phylogénie génétique, la matrice des distances est une matrice numérique qui indique à quel point chaque paire de séquences/clusters est similaire ou différente en termes d'évolution.

Matrice Q : Matrice numérique utilisé dans le contexte de la modélisation de la substitution moléculaire.

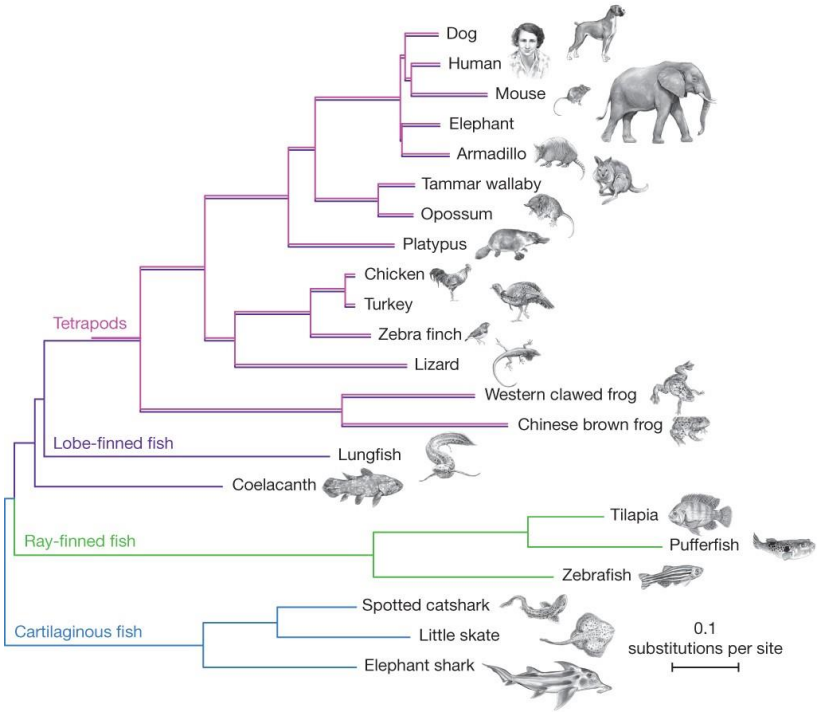


Figure 1 : Arbre phylogénétique des animaux

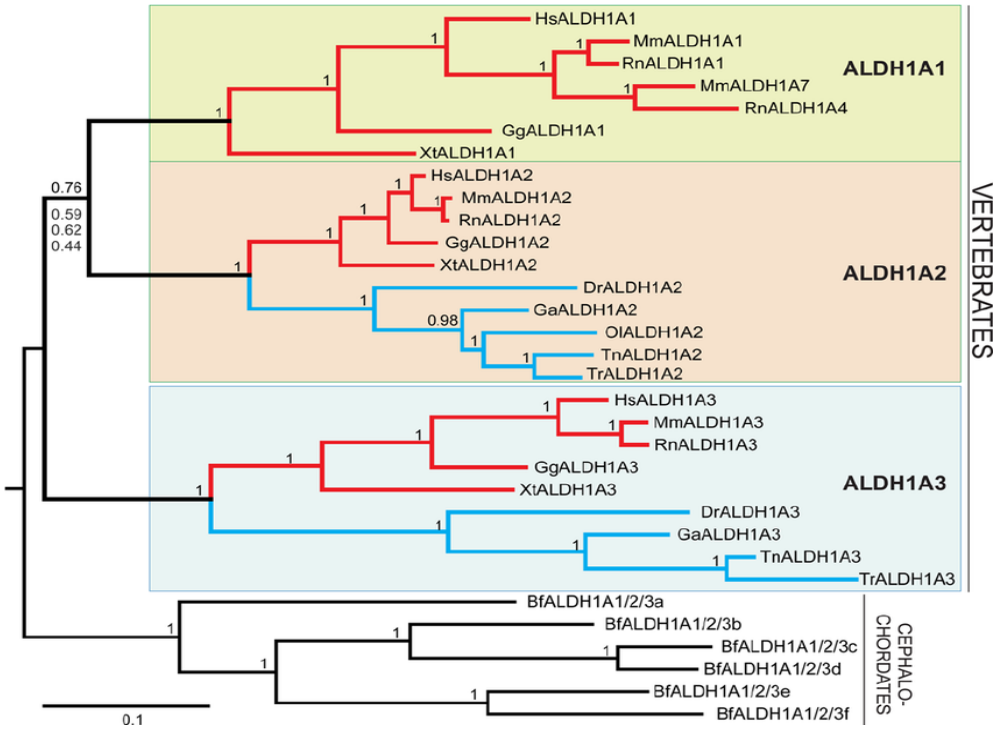


Figure 2 : Arbre phylogénétique de la famille de gènes Aldh1A chez les vertébrés

1. Introduction

1.1 La phylogénie

La phylogénie est le processus de reconstitution de l'histoire évolutive des êtres vivants à travers la création d'arbres phylogénétiques (Figure 1). Ces arbres montrent les relations de parenté entre espèces et indiquent quand elles ont divergé dans le temps. Les organismes sont positionnés sur l'arbre en fonction de leur similarité génétique, et des branches représentent ces liens, avec la longueur des branches indiquant le temps depuis le dernier ancêtre commun. Cela dépend de l'algorithme utilisé.

La phylogénie ne se limite pas aux organismes, elle inclut aussi les gènes qui subissent des mutations, créant de nouveaux phénotypes et contribuant à l'évolution. Par exemple, la famille de gènes Aldh1A chez les vertébrés illustre comment la phylogénie met en lumière les relations évolutives basées sur les similitudes génétiques (Figure 2).

Aujourd'hui, la phylogénie permet de représenter les liens entre séquences d'ADN, protéines et caractéristiques transmises de génération en génération, grâce à des algorithmes comme UPGMA (Unweighted Pair Group Method with Arithmetic Mean) et NJ (Neighbor Joining). Ces méthodes aident à visualiser l'évolution et à comprendre les liens de parenté dans le règne du vivant.

1.2 Les algorithmes de reconstruction phylogénétique

1.2.1 L’algorithme UPGMA

L'algorithme UPGMA une méthode permettant de construire un arbre enraciné (dendrogramme) à partir d'une matrice des distances. Sa spécificité réside dans le fait qu’elle suit l’hypothèse de l'horloge moléculaire qui suppose que les taux de changement moléculaire sont constants entre toutes les espèces. Cela implique que toutes les branches sont équidistantes du nœuds de départ. UPGMA est aussi un algorithme de regroupement (clustering) fonctionnant de manière itérative.

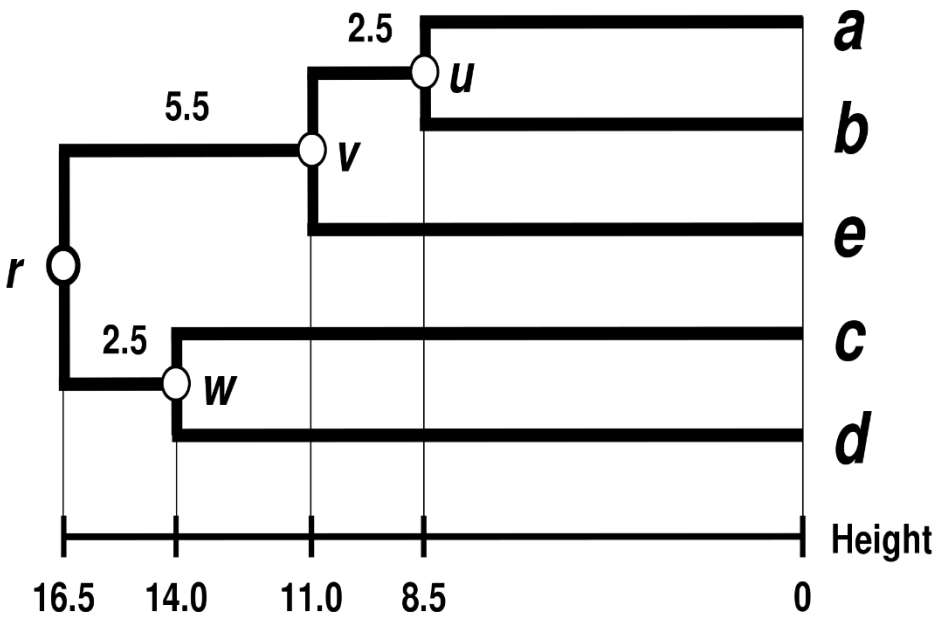


Figure 3 : Arbre phylogénétique obtenu avec
la méthode UPGMA

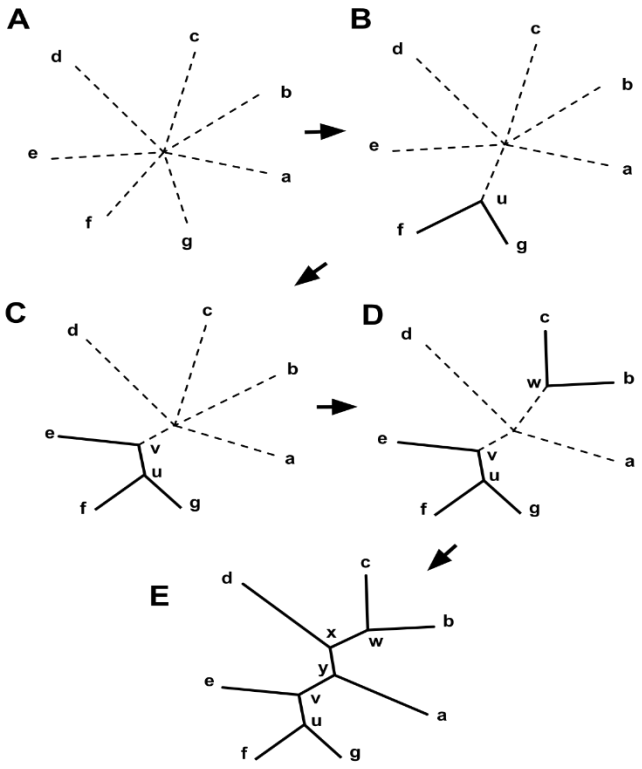


Figure 4 : Arbre phylogénétique obtenu avec
la méthode NJ

Voici comment il fonctionne :

1. Les séquences sont alignées les unes avec les autres, et le nombre de différence entre elles est compté et stocké dans une matrice des distances.
2. Deux séquences/clusters A et B présentant le moins de différences sont regroupés pour former un cluster AB.
3. Après la formation du cluster, les séquences/clusters sont alignés à nouveau et les distances moyennes entre le nouveau cluster et les autres séquences/clusters sont calculées. Les distances moyennes sont stockées dans une nouvelle matrice. Ici un exemple d'un calcul de distance moyenne pour un cluster AB et une séquence C :

$$\text{Distance}(AB, C) = ((\text{Distance}(A,C) + \text{Distance}(B,C)) / 2$$

Ces trois étapes sont répétées jusqu'à ce que la distance et le lien entre chaque séquence aient été calculés, permettant ainsi de construire l'arbre phylogénétique. La Figure 3 donne l'arbre obtenu à la fin sous forme d'un dendrogramme.

Il convient de noter que UPGMA n'est presque plus utilisé, car la théorie de l'horloge moléculaire est rarement confirmée. D'autres algorithmes ont été développés, ne présentant pas les mêmes défauts, tels que NJ (Neighbor Joining).

1.2.2 L'algorithme NJ

L'algorithme NJ représente une approche alternative en matière de regroupement dans le domaine de la phylogénétique. Contrairement à l'utilisation de l'horloge moléculaire, NJ prend en considération les disparités dans les taux d'évolution parmi les différentes branches de l'arbre phylogénétique qu'il tente de reconstruire. Son objectif principal est de préserver l'additivité des distances entre les espèces étudiées.

L'une des caractéristiques importantes de cette méthode est qu'elle produit un arbre phylogénétique non enraciné et non ultramétrique (Figure 4). Cela signifie que l'arbre résultant ne comporte pas de point de référence absolu (racine) et que les distances entre les espèces ne correspondent pas nécessairement à des intervalles de temps égaux. Il est tout de même possible d'obtenir un arbre sous forme de dendrogramme, pour faire des comparaisons avec d'autres

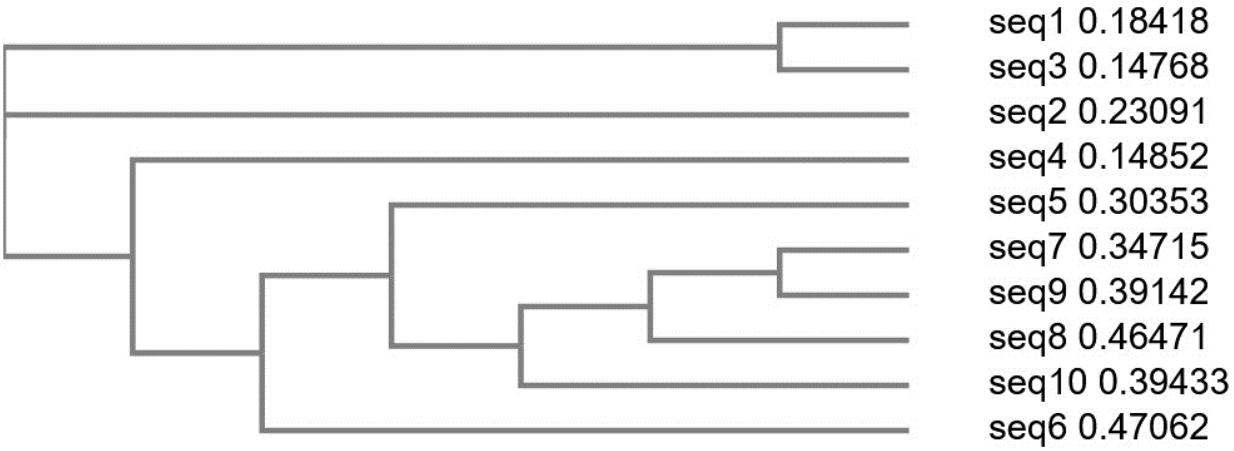


Figure 5 : Dendrogramme d'un arbre
obtenu avec la méthode NJ

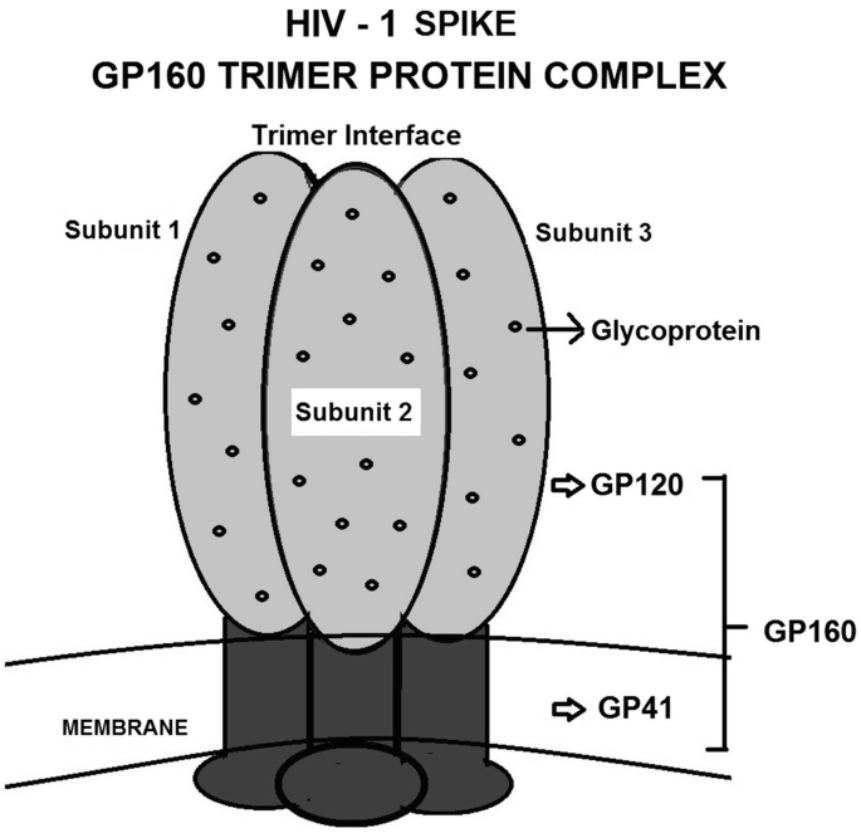


Figure 6 : La protéine Env/gp160 du VIH-1

arbres représentés de cette manière (Figure 5). L'algorithme, comme UPGMA, fonctionne de manière itérative avec une matrice des distances. Voici comment il fonctionne :

1. Calcul de la divergence nette de chaque séquence (ri) par rapport aux autres séquences (somme des différences d'une séquence avec toutes les autres séquences)
2. Remplir une matrice Q des distances modifiées avec la formule :

   Qij = Distance(i, j) - (ri + rj)/N-2

3. Prendre les séquences/clusters A et B avec la distance modifiée la plus petite et former un cluster AB
4. Calcul de la distance de chacune de deux séquences/clusters au nœud u avec la formule :

   Distance(A, u)= (Distance(A, B)/2) + (ra – rb)/2(N-2)

   Distance(B, u) = Distance(A, B) – Distance(A, u)

5. Calcul des distances u avec les autres nœuds et remplir la nouvelle matrice

Ces étapes se répètent jusqu'à que la matrice des distances soit vide.

1.3 La protéine Env du VIH/VIS

La protéine Env (ou gp160) constitue l'enveloppe du VIH/VIS (Figure 6). Elle joue un rôle essentiel dans le cycle de vie du virus en permettant sa pénétration dans les cellules cibles. Cette protéine se lie à des récepteurs spécifiques de la cellule hôte, favorisant ainsi la fusion du virus avec la cellule. De plus, elle est impliquée dans l'évasion du virus face au système immunitaire car elle présente une grande variabilité génétique. La protéine Env est synthétisée sous forme de précurseur, puis clivée par des enzymes en deux parties distinctes : gp120 et gp41, devenant ainsi fonctionnelle.

L'étude phylogénétique de cette protéine s'avère pertinente pour analyser son évolution au sein des différentes souches et virus du VIH/VIS.

L'objectif principal de ce projet est de développer un logiciel qui, dans un premier temps, prend en entrée des séquences ADN ou protéiques et génère un arbre phylogénétique à l'aide des méthodes UPGMA ou NJ, tout en proposant une interface graphique conviviale. Dans un second temps, le logiciel sera appliqué aux séquences protéiques et nucléiques de l'enveloppe Env, ainsi qu'à la séquence protéique Tat (Trans-Activator of Transcription), issues

```java
public class Method {
    protected int NumSeq;
    protected Sequence[] SequenceList;
    protected ArrayList<ArrayList<Double>> DistanceMatrix;

    public Method(String[] sequences) {
        NumSeq = sequences.length;
        DistanceMatrix = new ArrayList<>();
        SequenceList = new Sequence[NumSeq];
        for (int i = 0; i < NumSeq; i++) {
            //create branch for each sequence
            SequenceList[i] = new Sequence(sequences[i], i);
            //create the distance matrix
            DistanceMatrix.add(new ArrayList<Double>());
        }
        // initialize the distance matrix
        for (int i = 0; i < NumSeq; i++) {
            for (int j = 0; j < NumSeq; j++) {
                String sequence1 = sequences[i].replaceAll("[\n]", ""); //take out newline caracters
                String sequence2 = sequences[j].replaceAll("[\n]", "");
                DistanceMatrix.get(i).add(countMismatch(sequence1, sequence2, i, j));
                }
            if(DistanceMatrix.get(i).isEmpty()) {
                DistanceMatrix.remove(i);
            }
        }
        for (int i = 0; i < DistanceMatrix.size(); i++) {
            System.out.print(DistanceMatrix.get(i));
            System.out.println();
        }
    }
```

Figure 7 : Code partiel de la classe « Method »

de diverses souches du VIH/VIS.

Les séquences Env d'intérêt comprennent celles des souches A, B, C, D, N et O du VIH-1, des souches A et B du VIH-2, et des virus VIS du macaque (VISmac) et du chimpanzé (VIScpz).

2. Matériels et méthodes

2.1 Les séquences Env et Tat

Les séquences Env (environ 850 résidus) ont été extraites depuis la base de données protéiques Uniprot au format FASTA. Les sous-séquences correspondant aux protéines gp120 et gp41 d'Env pour chacun de ces virus, ainsi que leurs séquences de protéines Tat associées, ont été extraites aussi. Finalement, les séquences d'ADN Env des gènes de ces mêmes virus ont été récupérées grâce à une rétrotraduction faites sur le site SMS (Sequence Manipulation Suite).

2.2 Le logiciel de phylogénie

Le logiciel a été codé en Java (Annexe 1). Il prend en argument les fichiers FASTA des séquences à analyser. Il est constitué de 3 packages, chacun composé de 3 classes :

- Le package « Main » qui comprend la classe « Project », qui permet l'exécution du programme, l'affichage du GUI et le choix des méthodes. Il contient également la classe « MultipleFilesArgs », qui gère la lecture des fichiers d'entrée, ainsi que la classe « MethodGUI », qui permet l'affichage des arbres UPGMA et NJ.

- Le package « BuildTree » est dédié à la construction de l'arbre phylogénétique. La classe « Sequence » est utilisée pour créer les feuilles de l'arbre (nœuds externes), correspondant aux séquences. La classe « TreeNode » permet de créer les nœuds internes de l'arbre et de les relier aux séquences ou à d'autres nœuds. Enfin, la classe « NJTree » est spécifiquement conçue pour la construction de l'arbre selon la méthode NJ."

-Le package « Algorithms » permet la construction des arbres en utilisant l'algorithme UPGMA ou NJ. Il comprend la classe mère « Method » (Figure 7) et ses deux classes filles « UPGMA » et « NJ ». Lorsque l'une des deux classes filles est invoquée, la matrice des distances est initialisée dans « Method » en appelant la méthode « countMismatch » qui compte le nombre de différences entre une séquence et une autre. Les séquences sont stockées dans un tableau « SequenceList », sous forme d'objets de type « Sequence ».

```java
public class UPGMA extends Method {
    private static TreeNode[] Clusters;
    private String FinishedTree;


    public UPGMA(String[] sequences) {
        super(sequences);
        Clusters = new TreeNode[NumSeq]; //allows the same merges than the distance matrix with nodes and sequence branches
        double[] clusterSize = new double[DistanceMatrix.size()];
        for(int i = 0; i<clusterSize.length; i++) {
            clusterSize[i] = 1;
        }
        System.out.println("---------------------------------");
      //loop and update of the distance matrix and tree
        while(!isMatrixFinished(DistanceMatrix)) {
            int[] PairToMerge = smallestMismatchPair(DistanceMatrix);
            int index1 = PairToMerge[0];
            int index2 = PairToMerge[1];
            double NodeLength = returnTotalNodeLength(index1, index2, DistanceMatrix);
            DistanceMatrix = distanceMatrixUpdate(DistanceMatrix, index1, index2, clusterSize);
            mergeClusters(NodeLength, clusterSize, index1, index2);
            clusterSize[index1] = clusterSize[index1] + clusterSize[index2];
            for (int i = 0; i < DistanceMatrix.size(); i++) {
                System.out.print(DistanceMatrix.get(i));
                System.out.println();
            }
            System.out.println("---------------------------------");
        }
        TreeNode CompleteTree = Clusters[findIndexOfClusterMaxTaxa(Clusters)];
        ArrayList<Double> trueLengths = new ArrayList<>();
        getTrueNodeLength(CompleteTree, trueLengths);
        setTrueNodesLength(CompleteTree, trueLengths);
        CompleteTree.setLength(0.0); //set root node to 0.0
        //replace first node name with "root"
        String Tree = CompleteTree.toString();
        String[] lines = Tree.split("\n");
        lines[0] = lines[0].replaceFirst("^Node", "Root");
        FinishedTree = String.join("\n", lines);
    }
```

Figure 8 : Code partiel de la classe « UPGMA »

2.2.1 UPGMA

La classe « UPGMA » a été construite de la manière suivante (Figure 8). Elle commence d'abords avec l'initiation du tableau « Clusters », pouvant contenir des objets « TreeNodes », et la création du tableau « clusterSize » dont la valeur de chaque indice est de 1. La taille de ces tableaux est égale au nombre de séquences données en entrée.

Avec une boucle, les étapes suivantes se répètent jusqu'à que la matrice des distances soit remplie de 0.0 (vide) :

1) Les indices i et j correspondant à la plus petite valeur dans la matrice des distances sont trouvés à l'aide de la méthode « SmallestMismatchPair ». i/j peuvent correspondre soit à une séquence dans « SequenceList » soit à un cluster dans « Clusters ».
2) La longueur total des branches reliant les séquences/clusters à leur nœud père est calculée.
3) La matrice des distances est mise à jour en remplaçant toutes les valeurs de l'indice j par 0.0. Les valeurs dans la ligne i de la matrice, qui correspond maintenant à un nouveau cluster formé, sont remplacées par les nouvelles valeurs correspondant à la distance du cluster par rapport aux autres séquences/clusters. Ces nouvelles valeurs remplacent également la valeur à la colonne i de chaque ligne qui suit. La méthode appelée est « distanceMatrixUpdate ».
4) La méthode « mergeClusters » est appelée pour créer un nœud et le relier aux séquences/clusters. Pour déterminer si un index i/j correspond à une séquence ou un cluster, la valeur à l'indice i/j dans « clusterSize » est regardée. Si elle est égale à 1, cela signifie qu'elle correspond à une séquence. Par conséquent, la séquence située à l'index i/j dans « SequenceList » est reliée au nœud. Dans le cas contraire, c'est le cluster à l'index i/j dans « Clusters » qui est relié au nœud. Ce nouveau nœud est ensuite placé à l'indice i de « Clusters » pour simuler la mise à jour de la matrice. La longueur des branches est temporairement attribuée à ce nœud.
5) La valeur à l'indice i de « clusterSize » est mise à jour en l'additionnant avec la valeur à l'indice j.

Après que la matrice soit vide, la méthode « findIndexOfClusterMaxTaxa » est appliqué à « clusterSize » et renvoie l'indice du cluster le plus grand, soit l'indice correspondant à l'arbre dans son intégralité dans « Clusters ». Les longueurs des nœuds et feuilles de l'arbre sont ensuite mises à jour. Afin d'obtenir un arbre attribuant à chaque nœud/feuille la longueur de sa

```java
public class NJ extends Method {
    private String FinishedTree;

    public NJ(String[] sequences) {
        super(sequences);
        int maxNodes = SequenceList.length - 2; //to not create any more nodes by the end of the algorithm
        //create tree
        NJTree tree = new NJTree(SequenceList);
        //get cluster size for each clusters, to know if lone branch or cluster
        double[] clusterSize = new double[DistanceMatrix.size()];
        for(int i = 0; i<clusterSize.length; i++) {
            clusterSize[i] = 1;
        }
        double numSeq = DistanceMatrix.size();
        double[] Sums = new double[DistanceMatrix.size()];
        System.out.println();
        //loop and update of the distance matrix and tree
        while(!isMatrixFinished(DistanceMatrix)) {
            for (int i = 0; i<DistanceMatrix.size(); i++) {
                double SumOfSeq = 0;
                for (int j = 0; j<DistanceMatrix.get(i).size(); j++) {
                    SumOfSeq = SumOfSeq + DistanceMatrix.get(i).get(j);
                }
                Sums[i] = SumOfSeq;
            }
            ArrayList<ArrayList<Double>> Qmatrix = new ArrayList<>();
            for (ArrayList<Double> row : DistanceMatrix) {
                ArrayList<Double> newRow = new ArrayList<>(row);
                Qmatrix.add(newRow);
            }
            Qmatrix = calcQmatrix(numSeq, Qmatrix, Sums);
            for (int i = 0; i < Qmatrix.size(); i++) {
                System.out.print(Qmatrix.get(i));
                System.out.println();
            }
            System.out.println("---------------------------------");
            int[] PairToMerge = smallestMismatchPair(Qmatrix);
            int index1 = PairToMerge[0];
            int index2 = PairToMerge[1];
            double[] lengths = branchLengths(numSeq, DistanceMatrix, index1, index2, Sums);
            DistanceMatrix = distanceMatrixUpdate(DistanceMatrix, index1, index2);
            mergeClusters(clusterSize, index1, index2, lengths, tree, maxNodes);
            clusterSize[index1] = clusterSize[index1] + clusterSize[index2];
            for (int i = 0; i < DistanceMatrix.size(); i++) {
                System.out.print(DistanceMatrix.get(i));
                System.out.println();
            }
            System.out.println();
        }
        tree.getMainNode().setLength(0.0);
        FinishedTree = tree.getMainNode().toString();
    }
```

Figure 9 : Code partiel de la classe « NJ »

branche par rapport à son nœud père, les méthodes « getTrueNodeLength » et« setTrueNodeLength » ont été appliquées. La longueur initiale de chaque nœud est attribuée aux feuilles et sa nouvelle longueur est égal la longueur totale du nœud père moins sa longueur totale. L'arbre est ensuite stocké dans une variable « FinishedTree »

L'évolution de la matrice peut également être visualisée en sortie, ce qui permet de déterminer le nouveau cluster à chaque étape.

2.2.2 NJ

L'algorithme NJ, contrairement à UPGMA, part d'un arbre déjà établi qui doit être reconstruit. Pour respecter cette exigence, une classe nommée « NJTree » a été créée spécifiquement pour la méthode NJ. Lorsque la classe « NJ » est instanciée (Figure 9), elle crée une instance de « NJTree » et construit un arbre initial composé d'un nœud principal appelé « mainNode » qui est connecté à toutes les séquences.

Un tableau nommé « clusterSize » est créé, où chaque indice a une valeur de 1, de la même manière que pour « UPGMA ». De plus, un tableau appelé « Sums » est créé pour stocker les divergences nettes de chaque séquence.

Ces étapes sont répétées à l'aide d'une boucle jusqu'à ce que la matrice des distances soit remplie uniquement de 0.0 :

1) Calcul des divergences nettes et stockage des valeurs dans le tableau « Sums ».
2) Création et remplissage de la matrice Q.
3) Les indices i et j correspondant à la première valeur la plus petite dans la matrice Q sont retrouvés avec la méthode « SmallestMismatchPair », où i/j peuvent correspondre à une séquence dans « SequenceList » ou à un nœud dans le tableau « Nodes » de la classe « NJTree ».
4) Calcul de la distance de chacune des deux séquences/noeuds au nœud u.
5) Mise à jour de la matrice avec la méthode « distanceMatrixUpdate ». La méthode est pareille que celle dans « UPGMA » sauf que le calcul des nouvelles distances est différent.
6) Appelle à la méthode « mergeClusters ». La même logique que dans « UPGMA » est présente. La nature de i/j est indiqué dans le tableau « clusterSize ». L'instance de « NJTree » met à jour l'arbre. Si le nombre de nœuds est inférieur au nombre de

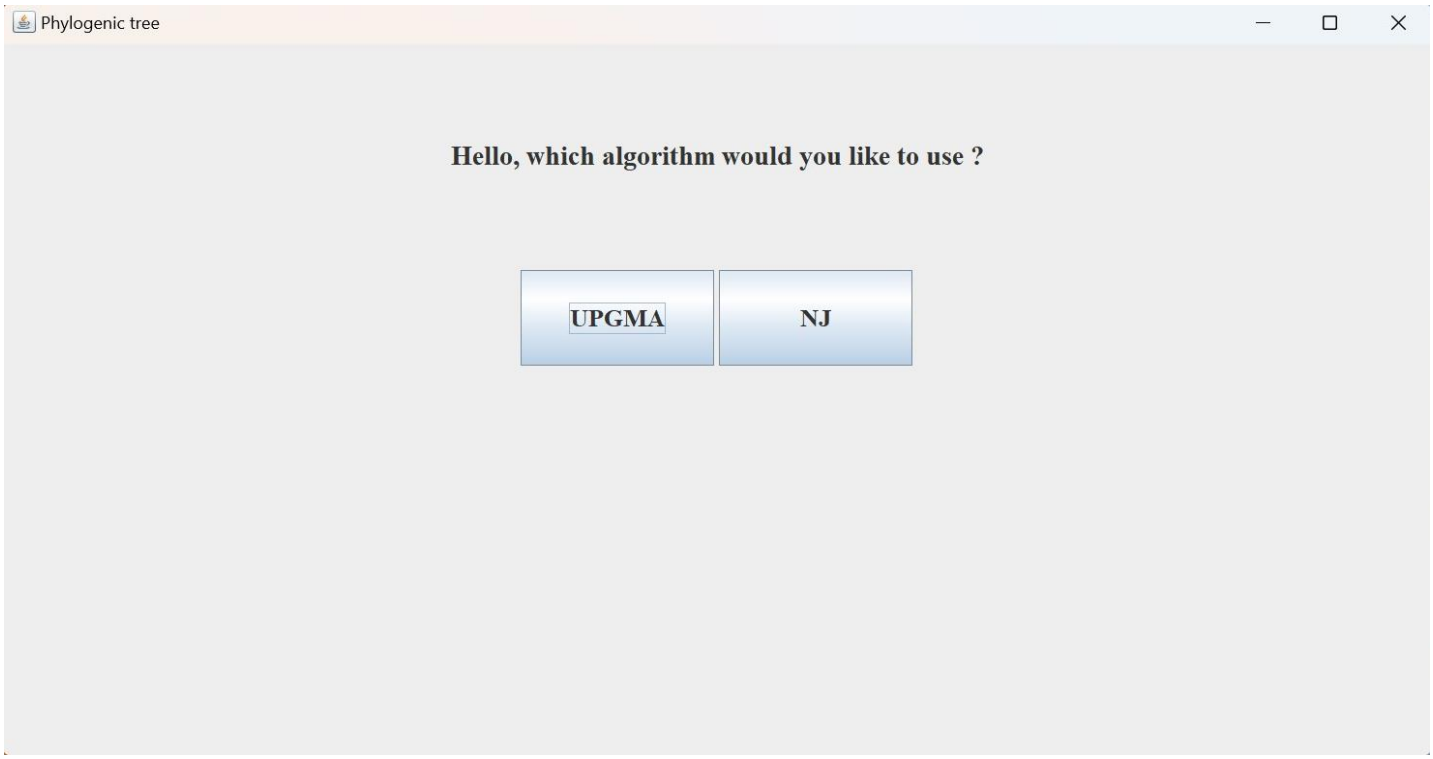


Figure 10 : Le GUI du programme

nœuds accepté (nombre de séquences – 2), un nouveau nœud est créé et relié aux séquences/nœuds auxquels des longueurs de branches sont attribuées. Ce nouveau nœud est ensuite relié au noeud « mainNode » de l'arbre. Les séquences/nœuds non reliés sont retirés de l'arbre. Dans le cas où le nombre maximum de nœuds est atteint, les longueurs des branches sont directement attribués aux séquences/nœuds correspondants.

7) La valeur à l'indice i de « clusterSize » est mise à jour en l'additionnant avec la valeur à l'indice j.

L'arbre complet, le « mainNode », est ensuite stocké dans une variable appelée « FinishedTree ». L'évolution de la matrice et de la matrice Q peut être visualisée en sortie.

2.2.3 GUI

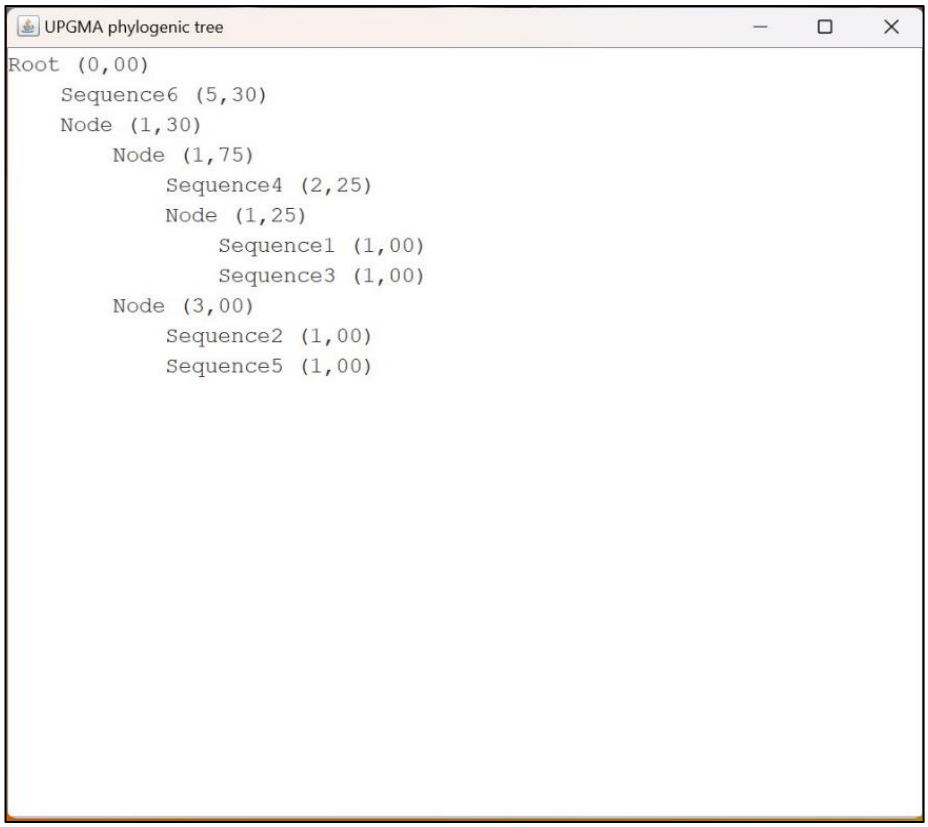
L'interface graphique, conçue en utilisant Java Swing, est intégrée dans le « main » du programme. Deux boutons, l'un représentant "UPGMA" et l'autre "NJ", sont positionnés au centre de l'interface. Lorsqu'un de ces boutons est activé par l'utilisateur, une instance de la méthode sélectionnée est instanciée. Ensuite, une instance de « MethodGUI » est créée, générant ainsi sa propre interface dédiée. Cette interface interagit avec la méthode en appelant la variable "FinishedTree".

3. Résultats

3.1 Lancement du logiciel de phylogénie

Lorsque le programme est lancé, une interface s'affiche avec les deux boutons « UPGMA » et « NJ » (Figure 10). Cliquer sur un des boutons fait apparaître l'arbre, suivant la méthode choisis, dans une autre interface. Les boutons peuvent être enclenchés sans faire l'autre interface disparaitre, ce qui offre la possibilité de comparer directement les arbres générés par les deux méthodes. Les séquences données en entrée doivent être à la fois distinctes et avoir le même nombre de caractère. Dans le cas contraire, une erreur se produira.

A



B

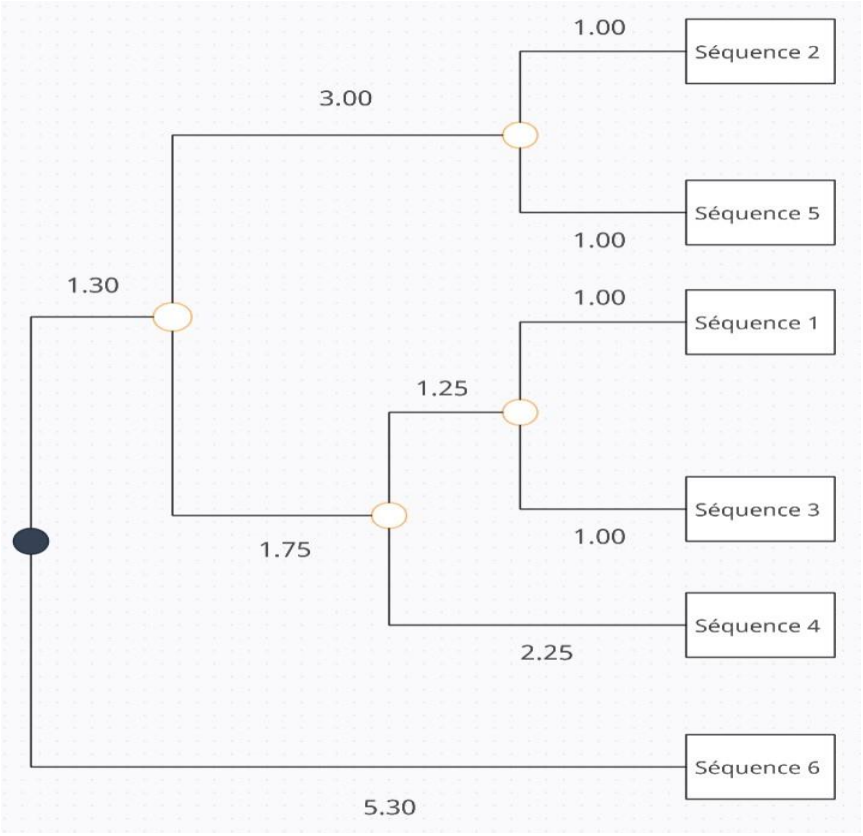


Figure 11 : Exemple d'un arbre UPGMA obtenu avec le programme (A) et de sa représentation en dendrogramme (B)

3.2 L’arbre phylogénétique du logiciel

L’arbre UPGMA obtenu (Figure 11.A) est présenté sous forme textuelle. Il utilise des indentations pour une meilleure lisibilité, permettant d'identifier les nœuds fils et les feuilles associés à chaque nœud. La longueur de la branche d’une feuille/nœud à son nœud père est affiché à côté. L'utilisateur a la possibilité de dessiner un dendrogramme de l'arbre pour faciliter l'analyse ultérieure (Figure 11.B). L’arbre NJ, quant à lui, est représenté de la même manière mais n’est pas enraciné (ne contient pas de racine « Root »).

3.3 Les arbres phylogénétiques Env et Tat du VIH/VIS

Le logiciel a été utilisé avec les deux options UPGMA et NJ pour les catégories suivantes : les séquences Env complètes, les tronçons Env 1-100, les tronçons Env 101-200, les tronçons Env 201-300, les tronçons Env 301-400, les protéines gp120, les protéines gp41, les protéines Tat, les séquences d'ADN Env 300-600, et les séquences d'ADN Env 600-900. Les séquences ont été soumises à un alignement multiple à l'aide de Clustal MSA (Multiple Sequence Alignment), ce qui a permis d'homogénéiser le nombre de caractères dans chaque séquence, ce qui est obligatoire pour faire fonctionner le programme, et d'améliorer la précision des arbres phylogénétiques résultants.

Pour simplifier l’analyse, les souches VIH/VIS ont reçu les noms suivants pour chaque arbre au lieu de leur numéro d’accession :

-VIH-1 souche A => Sequence1

-VIH-1 souche B => Sequence2

-VIH-1 souche C => Sequence3

-VIH-1 souche D => Sequence4

-VIH-1 groupe N => Sequence5

-VIH-1 groupe O => Sequence6

-VIH-2 souche A => Sequence7

-VIH-2 souche B => Sequence8

-VISmac => Sequence9

-VIScpz => Sequence10

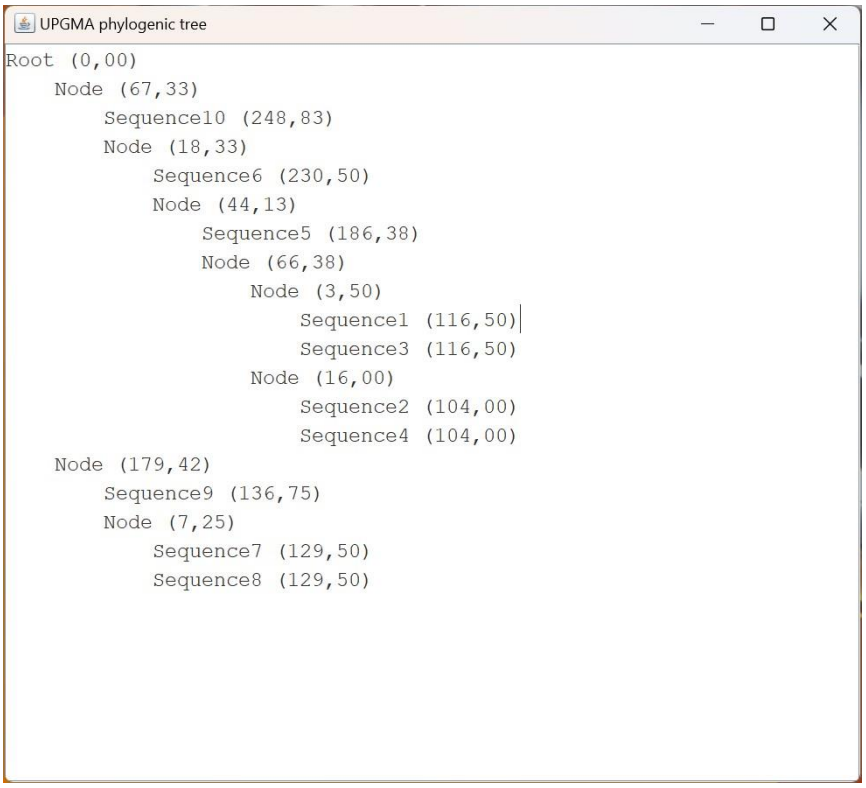


Figure 12 : Arbre UPGMA des séquences Env complètes

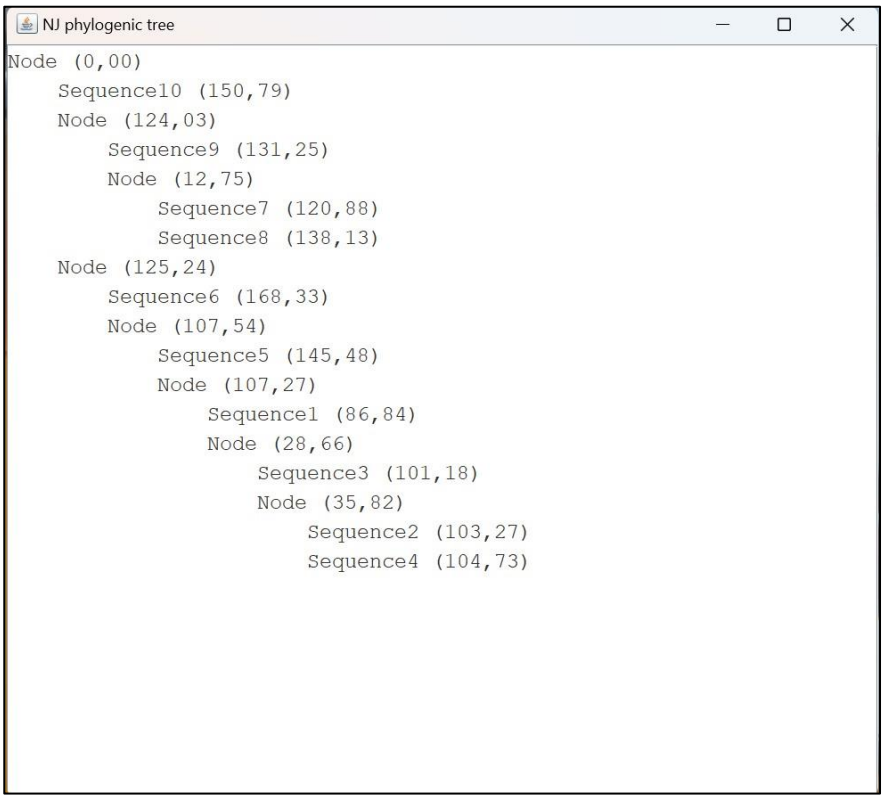


Figure 13 : Arbre NJ des séquences Env complètes

Les arbres des séquences Env complètes se trouvent en Figures 12 et 13, les tronçons Env 1-100 en Annexes 2 et 3, les tronçons Env 101-200 en Annexes 4 et 5, les tronçons Env 201-300 en Annexes 6 et 7, les tronçons Env 301-400 en Annexes 8 et 9, les protéines gp120 en Annexes 10 et 11, les protéines gp41 en Annexes 12 et 13, les protéines Tat en Annexes 14 et 15, les séquences d'ADN Env 300-600 en Annexes 16 et 17, et enfin les séquences d'ADN Env 600-900 en Annexes 18 et 19.

4. Discussion

4.1 Analyse des arbres

En examinant les arbres construits à partir des séquences protéiques ou nucléiques Env (gp160/gp120/gp41), un schéma commun est observé. Au début du processus évolutif, deux groupes majeurs ont émergé. Le premier groupe, Groupe 1, est composé du VIH-1 souche A, du VIH-1 souche B, du VIH-1 souche C et du VIH-1 souche D. Le deuxième groupe, Groupe 2, comprend le VIH-2 souche A, le VIH-2 souche B et le VISmac.

Dans l'ensemble, les divergences au sein de chaque groupe ne sont pas toujours identiques, mais elles demeurent cohérentes. Cette cohérence est observée à la fois dans les résultats obtenus par la méthode UPGMA et par la méthode NJ. Les variations se manifestent principalement dans la manière dont les séquences sont positionnées dans l'arbre, mais les groupes de séquences restent relativement stables dans leur composition.

Cependant, des différences substantielles entre les arbres UPGMA et NJ se manifestent dans le cas des séquences du VIH-1 groupe N, du VIH-1 groupe O et du VIScpz. Avec UPGMA, ces séquences sont généralement regroupées au sein du Groupe 1, présentant une divergence précoce au sein de ce groupe. En revanche, avec la méthode NJ, il est plus courant de constater qu'elles ne sont pas affiliées à un groupe majeur, ayant manifesté une divergence précoce et se trouvant isolées au sein de l'arbre évolutif.

Les arbres de la protéine Tat révèlent également la cohérence des groupes 1 et 2, à l'exception de la méthode UPGMA, où la séquence Tat du VIH-1 souche A a divergé précocement au sein du Groupe 1, contrairement aux séquences Env. Dans le cas de la méthode NJ, cette séquence n'est pas affiliée à un groupe majeur et a manifesté une divergence précoce.

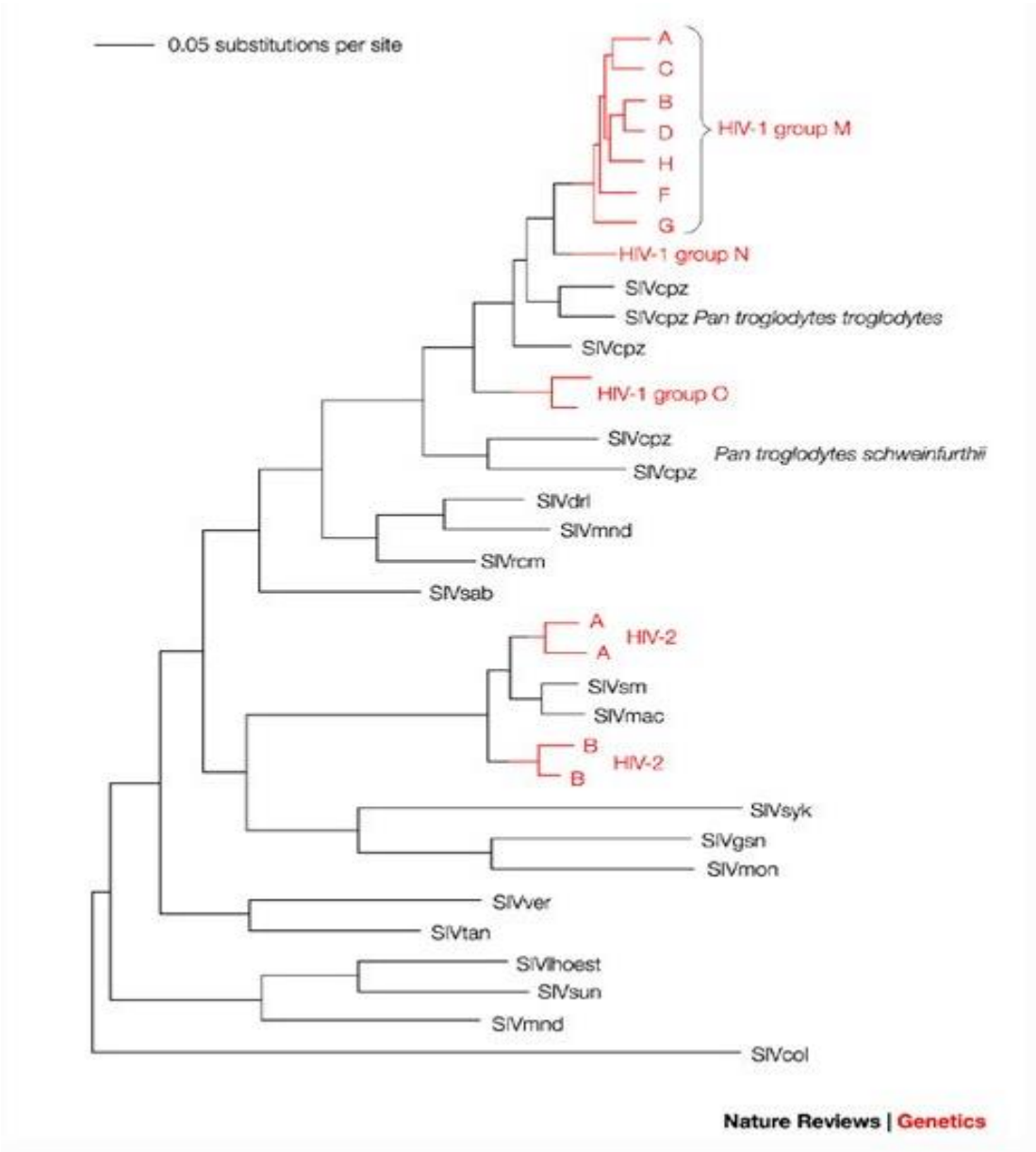


Figure 14 : Arbre phylogénétique des souches VIH/VIS

4.2 Cohérence des résultats du logiciel de phylogénie

Il est tout à fait cohérent de constater que les souches VIH-1 souche A, VIH-1 souche B, VIH-1 souche C et VIH-1 souche D sont regroupées dans le même groupe au sein de l'arbre phylogénétique car elles appartiennent toutes au groupe M du VIH-1, ce qui indique une parenté évolutive étroite entre elles. De manière similaire, il est logique de constater que les souches VIH-2 souche A et VIH-2 souche B sont très proches sur les arbres obtenus, leurs noms suggèrant une proximité évolutive entre elles.

La Figure 14 présente un arbre phylogénétique des souches du VIH/VIS, extrait de l'article intitulé « The causes and consequences of HIV evolution » publié dans la revue Nature. Dans cet arbre, les deux groupes principaux observés dans les arbres phylogénétiques générés par le logiciel sont identifiables. Cette figure confirme aussi le placement du VISmac à proximité du VIH-2, ce qui est cohérent avec nos observations précédentes.

Par ailleurs, cette représentation graphique permet également de mettre en évidence les divergences entre les méthodes UPGMA et NJ, en particulier en ce qui concerne les séquences du VIH-1 groupe N, du VIH-1 groupe O et du VIScpz. En effet, ces séquences ne sont pas regroupées dans l'un des deux groupes principaux de manière uniforme, malgré la proximité du VIH-1 groupe N avec le Groupe 1. Cela souligne les nuances dans la manière dont ces méthodes reconstruisent l'arbre phylogénétique.

5. Conclusion

Le programme élaboré au sein de ce projet, qui exploite les méthodes UPGMA et NJ, démontre de manière convaincante que ces deux approches permettent de générer des arbres phylogénétiques à partir des séquences Env et Tat qui présentent une cohérence remarquable. En effet, les deux groupes principaux, conformes à nos attentes, sont clairement identifiables et leur existence est confirmée par des sources externes.

Il serait désormais judicieux de procéder à la validation de ces arbres en utilisant plusieurs logiciels de phylogénie qui utilisent à la fois UPGMA et NJ. Cette validation aurait pour objectif de non seulement confirmer la structure des arbres obtenus, mais aussi d'évaluer la longueur des branches de chaque nœud de manière plus précise. De plus, il serait particulièrement intéressant de comparer les différences entre les résultats produits par ce

logiciel et ceux générés par d'autres logiciels de phylogénie. Cette démarche permettrait d'approfondir la compréhension des variations potentielles liées à l'utilisation de différentes méthodes et outils dans le domaine de la phylogénie.

6. Bibliographies

Figures

Figure 1 : https://plato.stanford.edu/entries/phylogenetic-inference/

Figure 2 : https://www.researchgate.net/figure/Phylogenetic-tree-of-the-vertebrate-Aldh1A-gene-family-All-phylogenetic-methodologies_fig1_26250098

Figure 3 : https://en.wikipedia.org/wiki/UPGMA#/media/File:UPGMA_Dendrogram_5S_data.svg

Figure 4 : https://en.wikipedia.org/wiki/Neighbor_joining#/media/File:Neighbor_joining_7_taxa_start_to_finish_diagram.svg

Figure 5 : https://www.ebi.ac.uk/Tools/services/web/toolresult.ebi?jobId=simple_phylogeny-I20230922-131354-0225-43583413-p1m

Figure 6 : https://link.springer.com/chapter/10.1007/978-3-319-95327-4_9

Figure 14 : Andrew Rambaut, David Posada, Keith A. Crandall & Edward C. Holmes (2004). "The causes and consequences of HIV evolution." Nature Reviews Genetics, 5, 52–61 https://www.nature.com/articles/nrg1246

Séquences Env/gp160, gp120, gp41

-HPV-1 Souche A : https://www.uniprot.org/uniprotkb/P05881/entry

-HPV-1 Souche B : https://www.uniprot.org/uniprotkb/P05877/entry

-HPV-1 Souche C : https://www.uniprot.org/uniprotkb/O12164/entry

-HPV-1 Souche D : https://www.uniprot.org/uniprotkb/P04581/entry

-HPV-1 Souche N : https://www.uniprot.org/uniprotkb/Q9IDV2/entry

-HPV-1 Souche O : https://www.uniprot.org/uniprotkb/Q79670/entry

-HPV-2 Souche A : https://www.uniprot.org/uniprotkb/P20872/entry

-HPV-2 Souche B : https://www.uniprot.org/uniprotkb/P15831/entry

-VISmac : https://www.uniprot.org/uniprotkb/P05885/entry

-VIScpz : https://www.uniprot.org/uniprotkb/Q8AIH5/entry

Séquences Tat

-HPV-1 Souche A : https://www.uniprot.org/uniprotkb/P12512/entry

-HPV-1 Souche B : https://www.uniprot.org/uniprotkb/P05905/entry

-HPV-1 Souche C : https://www.uniprot.org/uniprotkb/O12161/entry

-HPV-1 Souche D : https://www.uniprot.org/uniprotkb/P04611/entry

-HPV-1 Souche N : https://www.uniprot.org/uniprotkb/Q9IDV5/entry

-HPV-1 Souche O : https://www.uniprot.org/uniprotkb/P0C1K2/entry

-HPV-2 Souche A : https://www.uniprot.org/uniprotkb/P20880/entry

-HPV-2 Souche B : https://www.uniprot.org/uniprotkb/P15835/entry

-VISmac : https://www.uniprot.org/uniprotkb/P05911/entry

-VIScpz : https://www.uniprot.org/uniprotkb/Q8AIH8/entry

SMS : https://www.bioinformatics.org/sms2/rev_trans.html

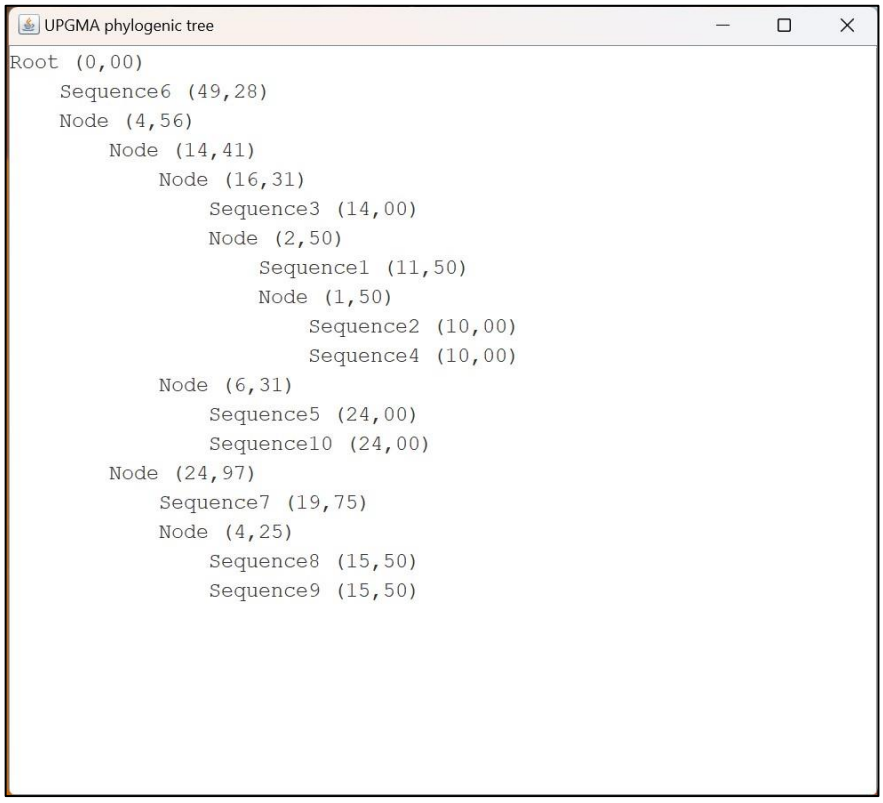
Clustal MSA : https://www.ebi.ac.uk/Tools/msa/clustalo/

7. Annexes

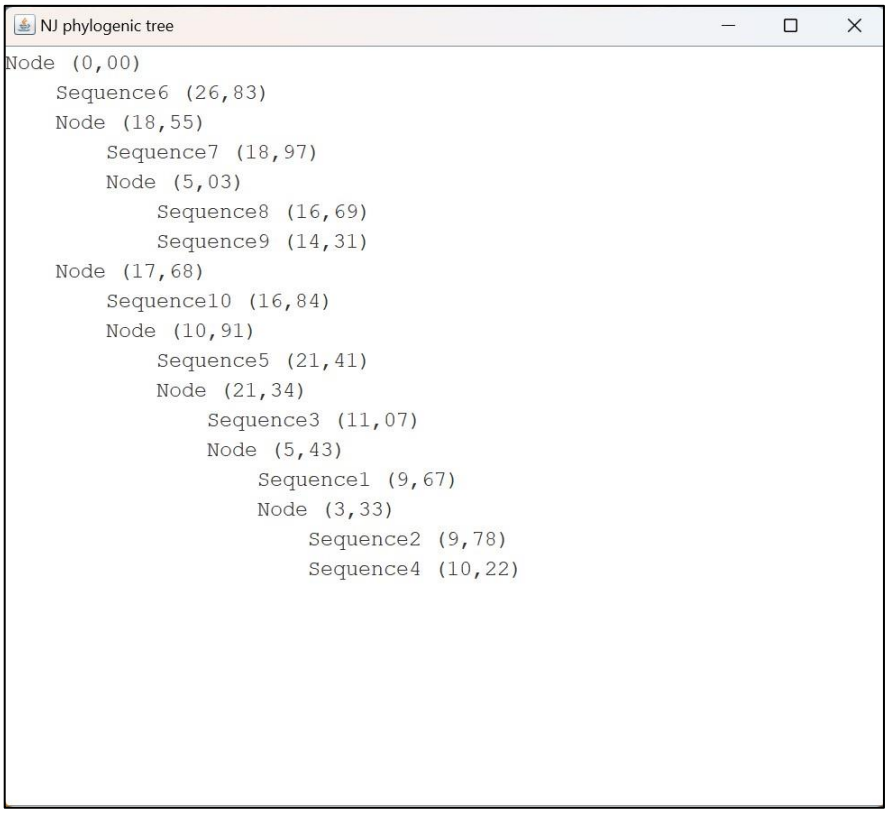
Annexe 1 : Code complet du logiciel de phylogénie

https://github.com/geap1999/PhylogeneticTreeProject/tree/main/PhyloTreeProject

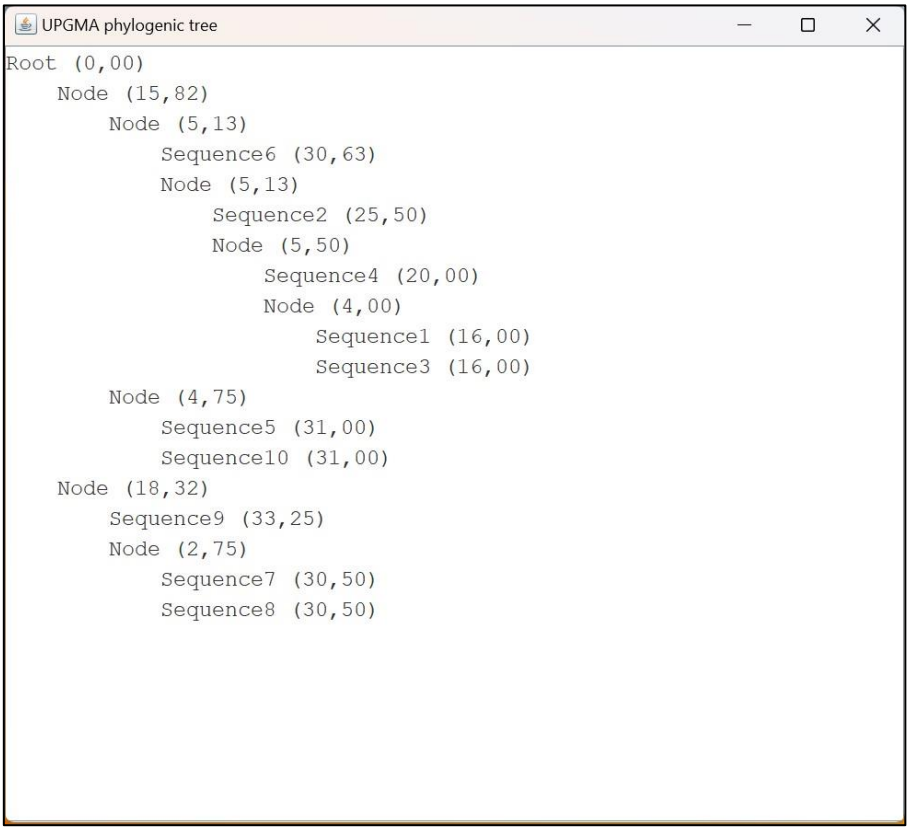
Annexe 2 : Arbre UPGMA pour les tronçons Env 1-100



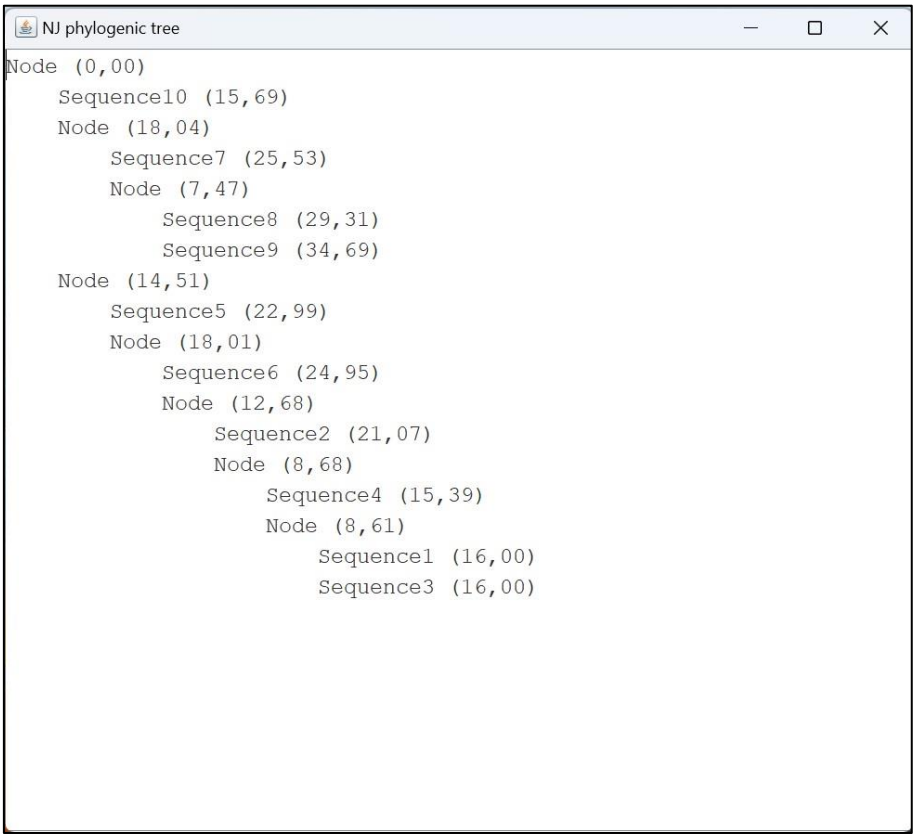
Annexe 3 : Arbre NJ pour les tronçons Env 1-100



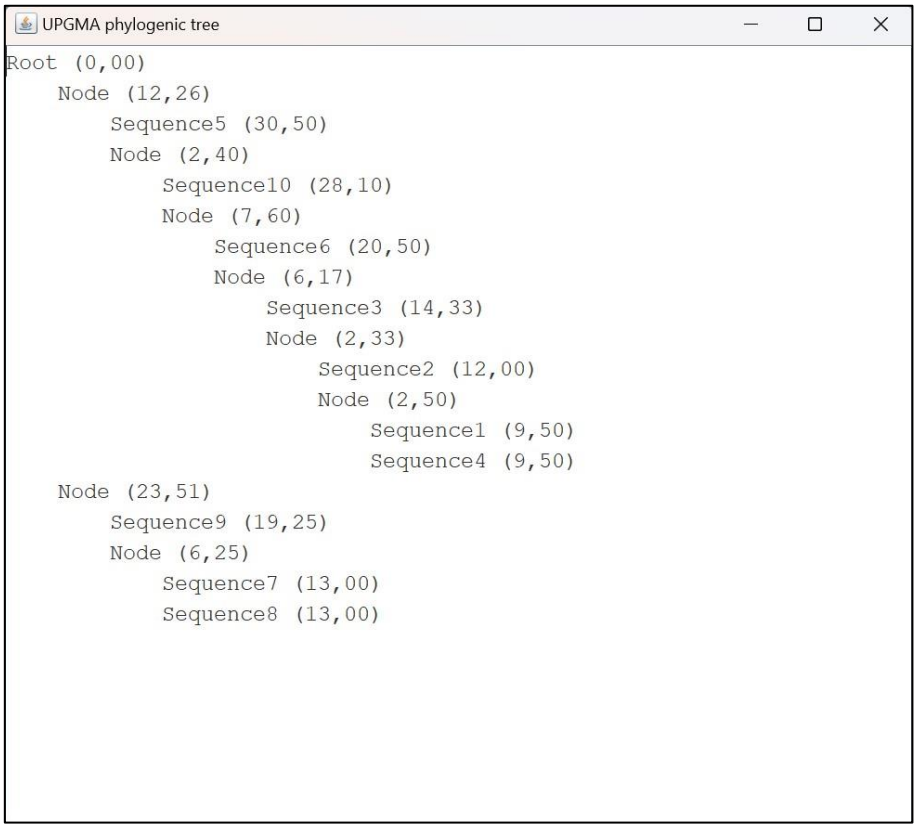
Annexe 4 : Arbre UPGMA pour les tronçons Env 101-200



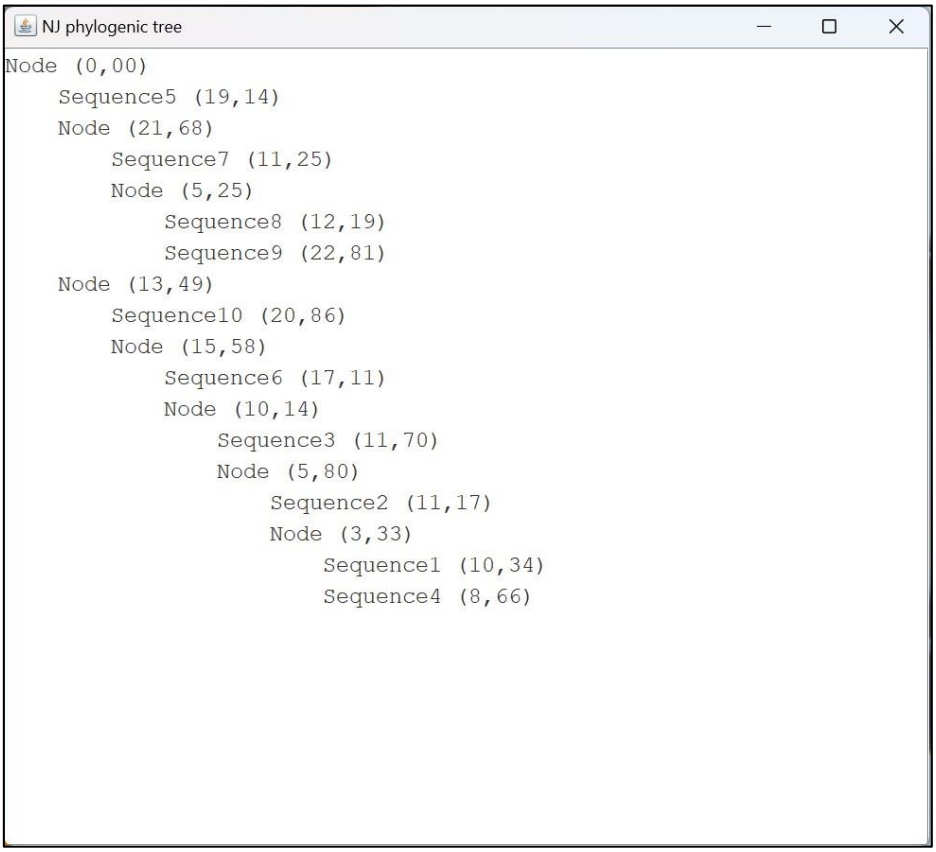
Annexe 5 : Arbre NJ pour les tronçons Env 101-200



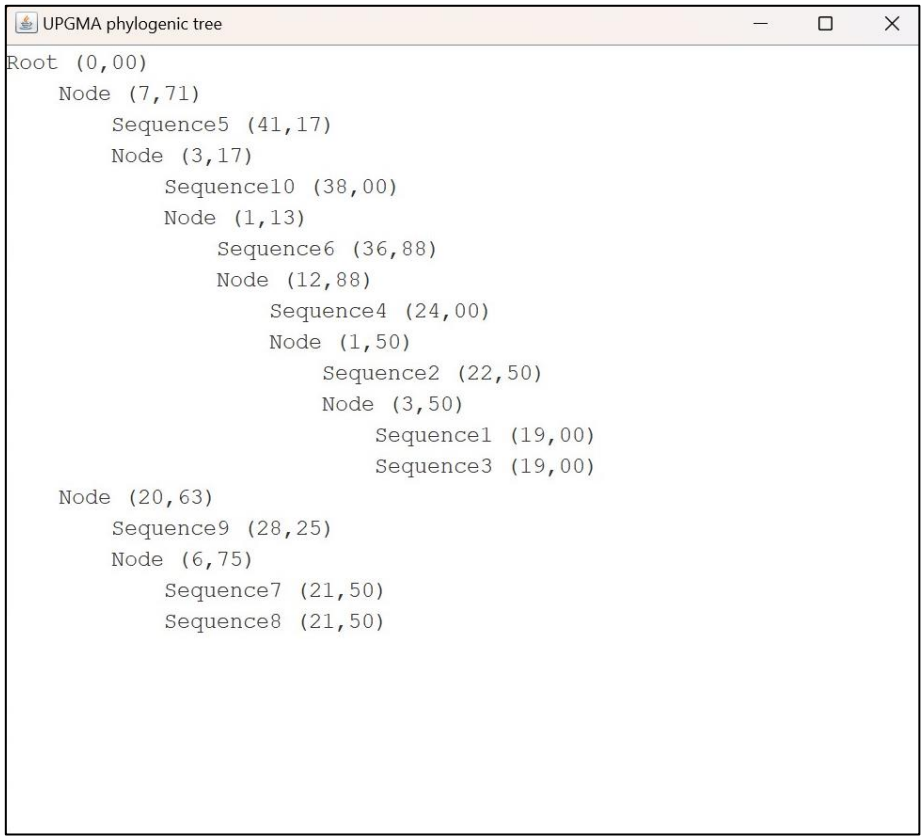
Annexe 6 : Arbre UPGMA pour les tronçons Env 201-300



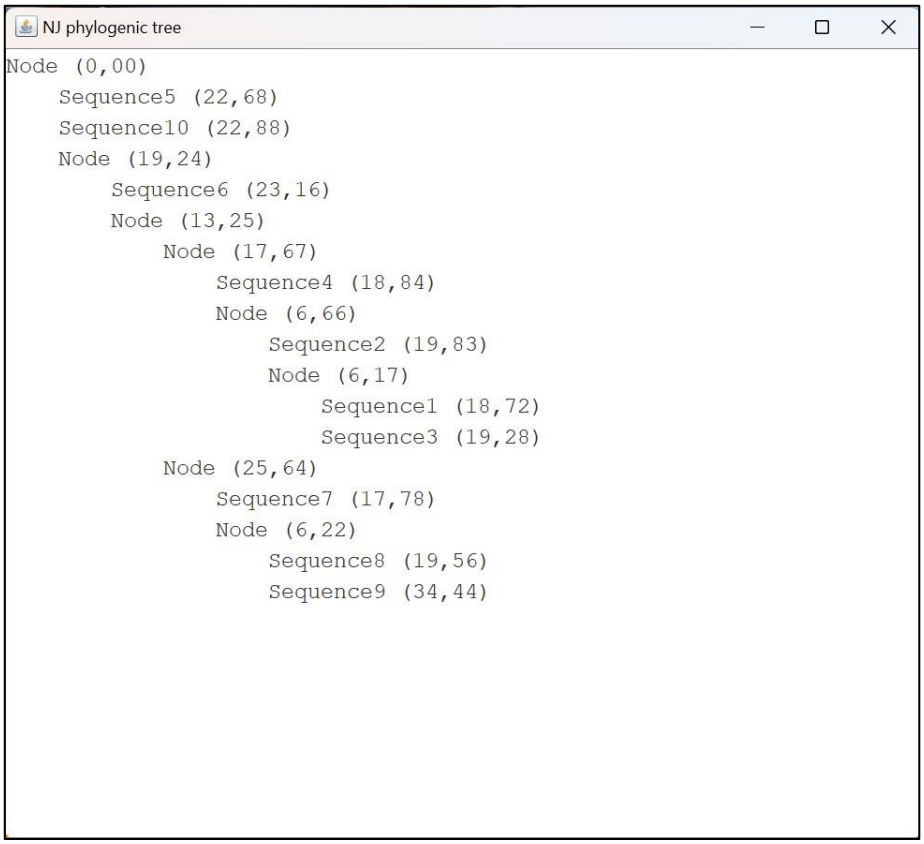
Annexe 7 : Arbre NJ pour les tronçons Env 201-300



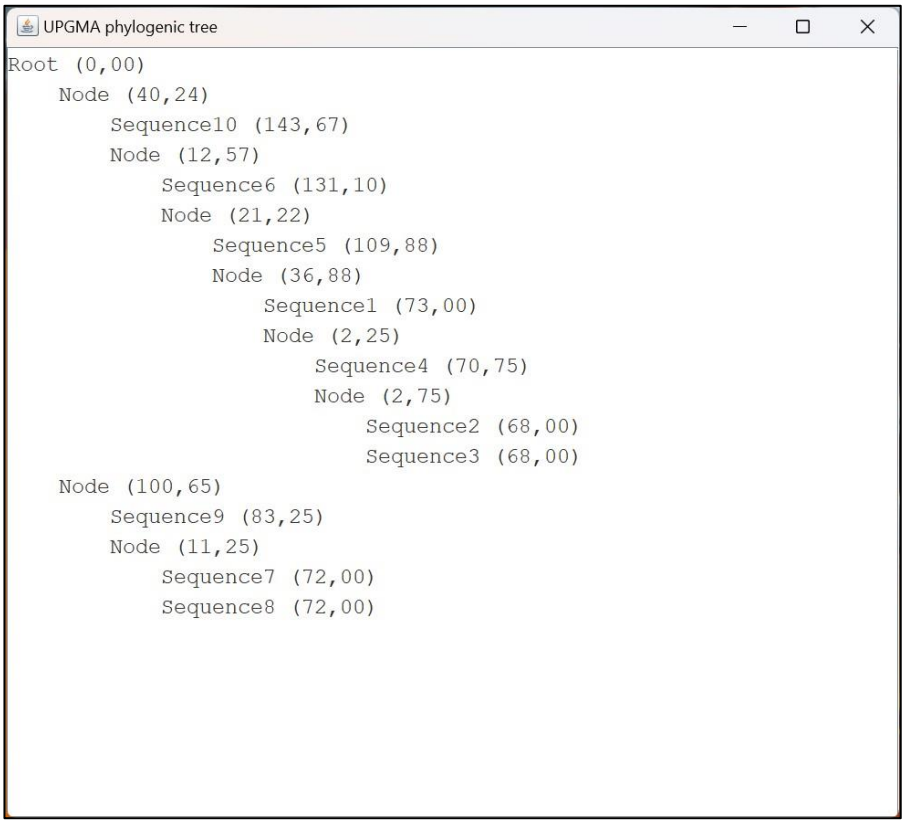
Annexe 8 : Arbre UPGMA pour les tronçons Env 301-400



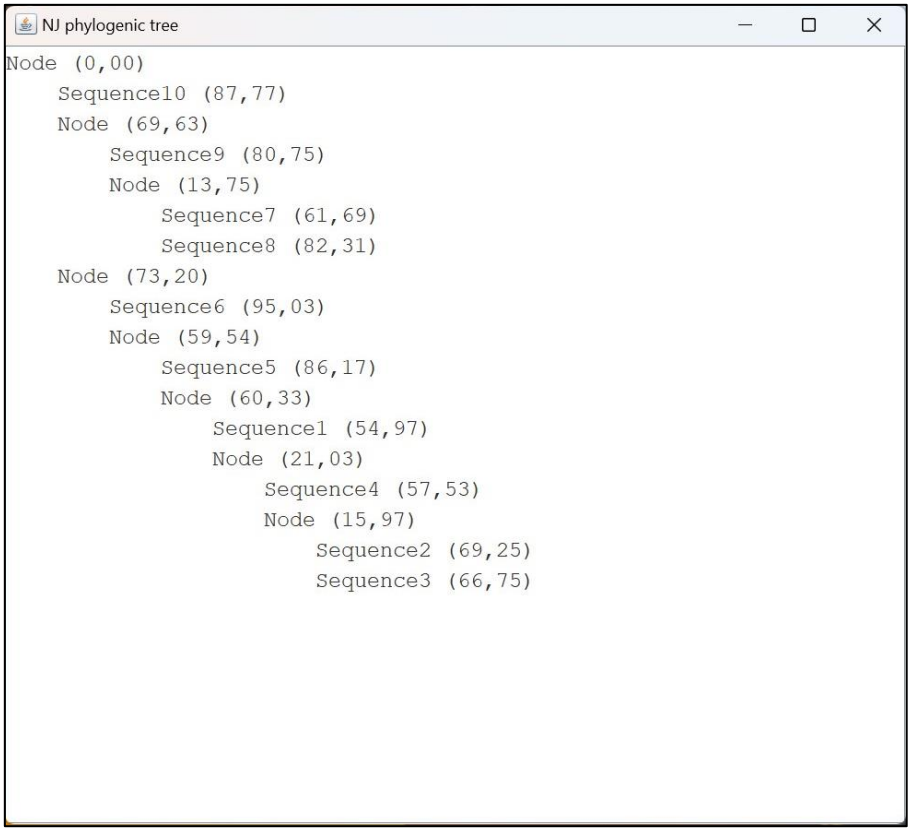
Annexe 9 : Arbre NJ pour les tronçons Env 301-400



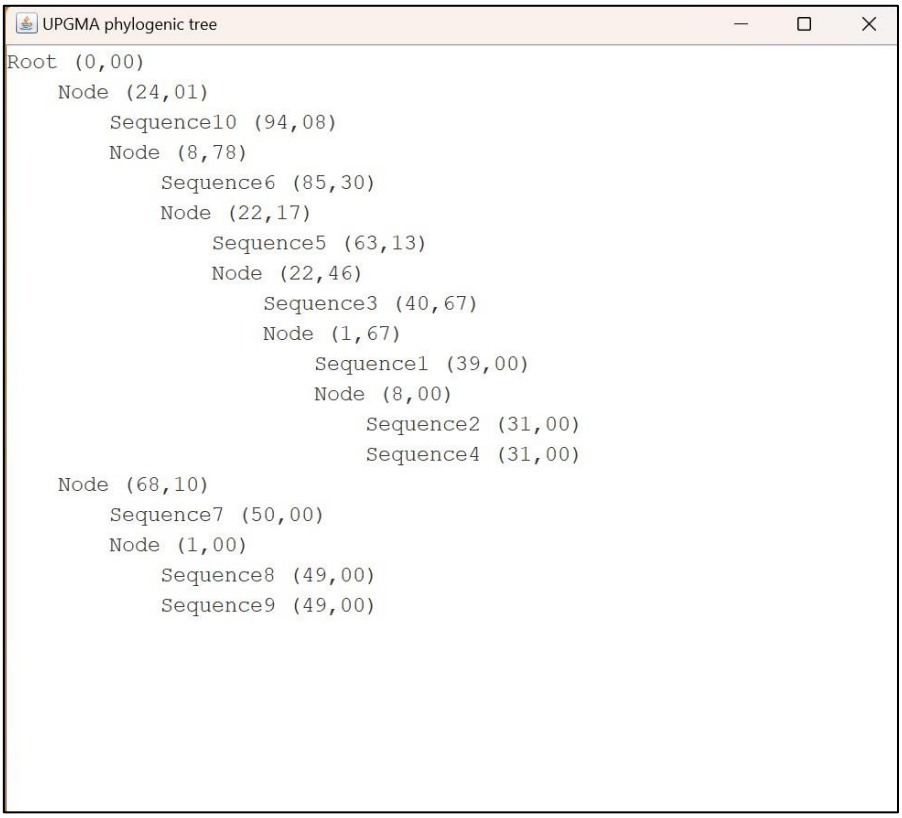
Annexe 10 : Arbre UPGMA pour la protéine gp120



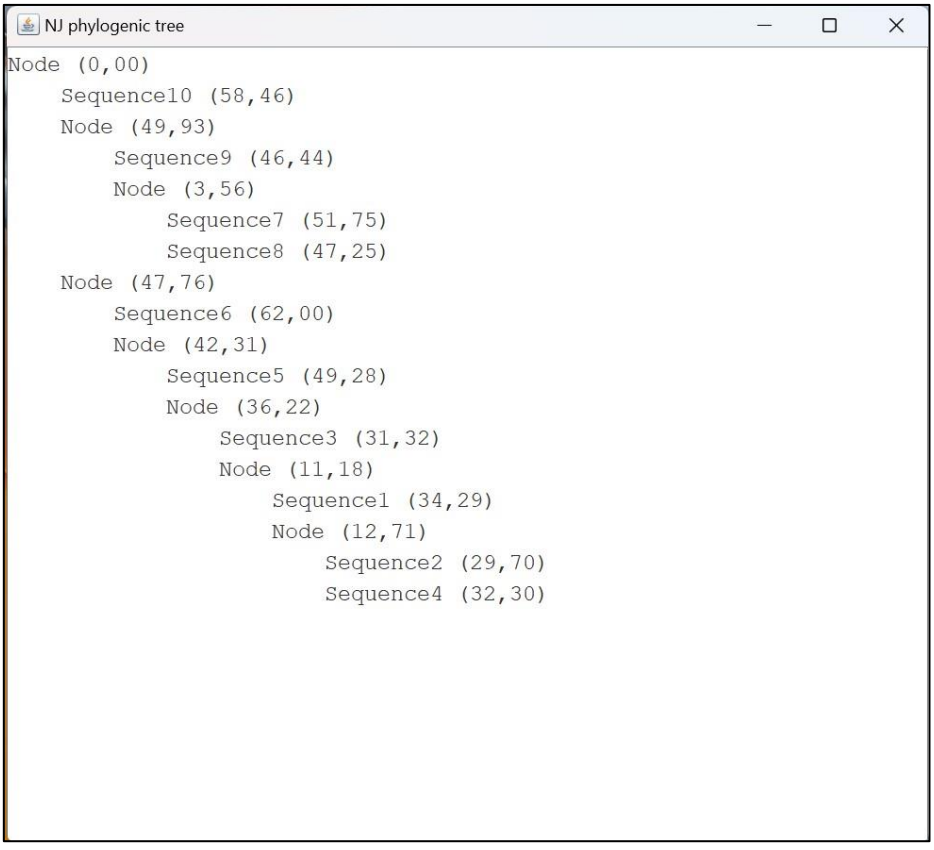
Annexe 11 : Arbre NJ pour la protéine gp120



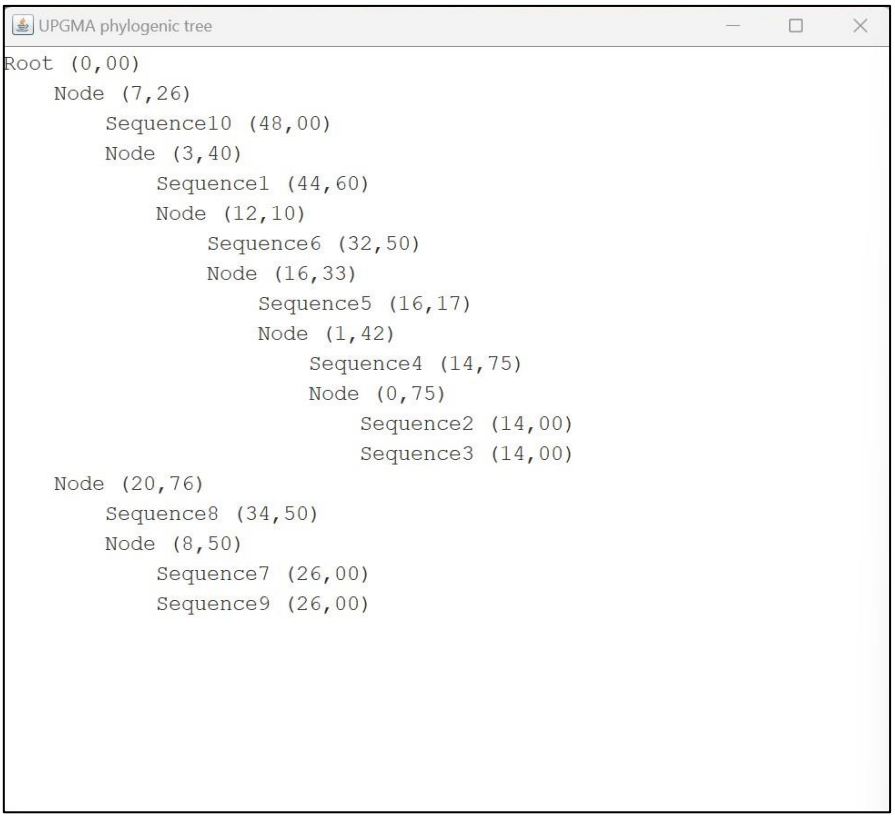
Annexe 12 : Arbre UPGMA pour la protéine gp41



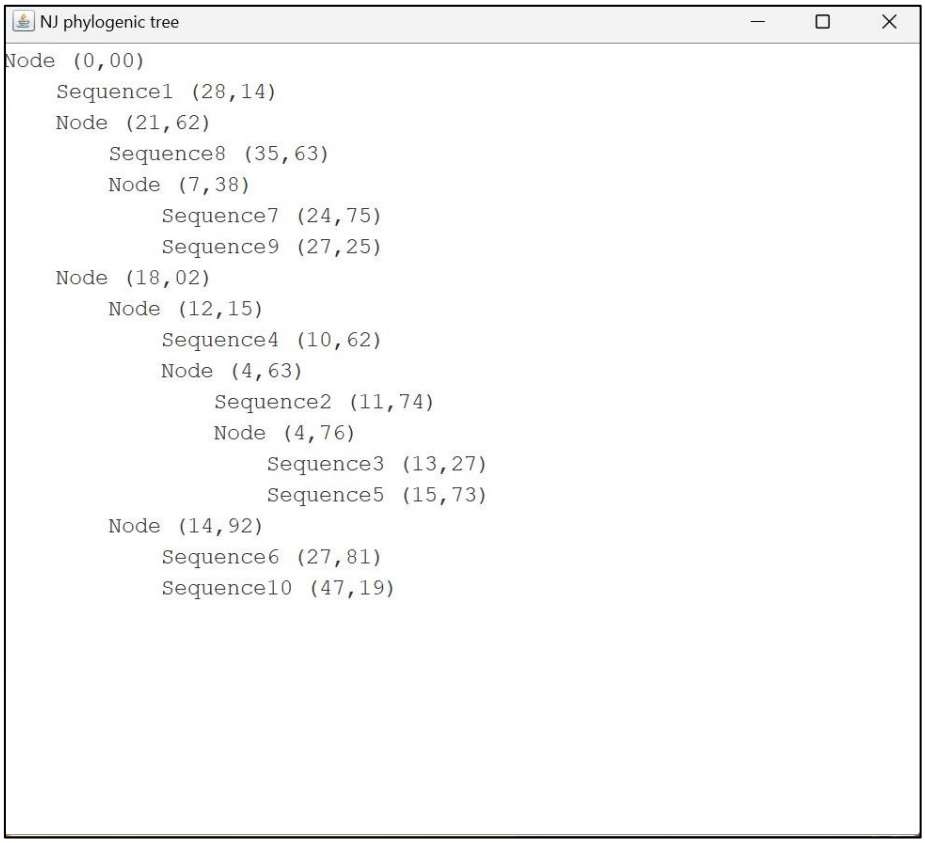
Annexe 13 : Arbre NJ pour la protéine gp41



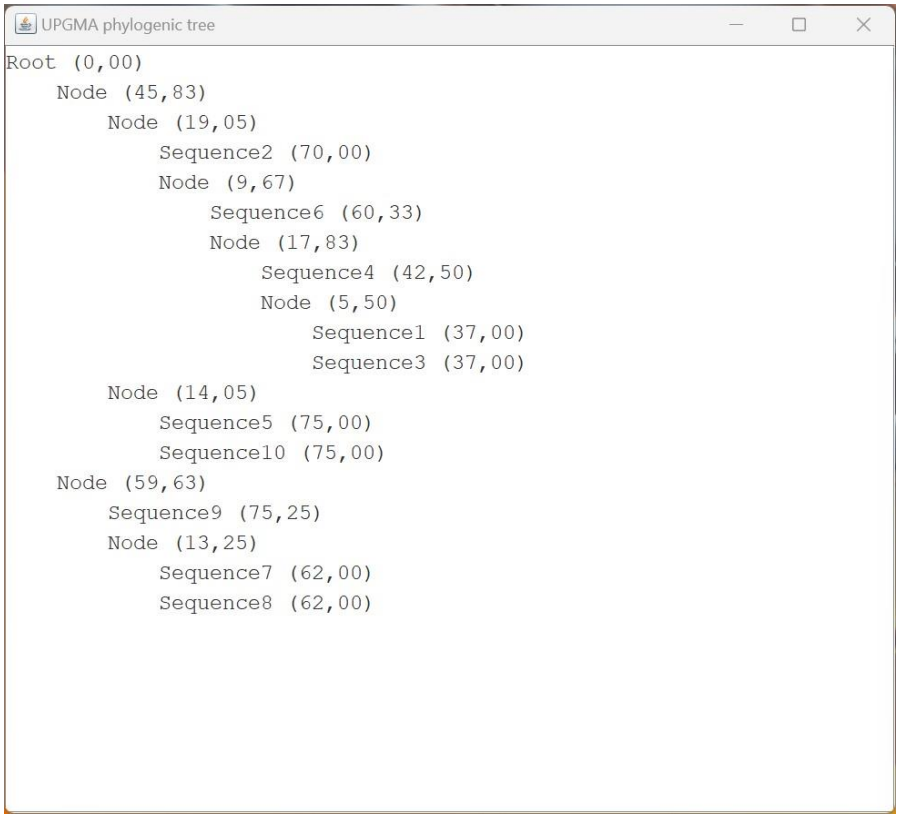
Annexe 14 : Arbre UPGMA pour la protéine Tat



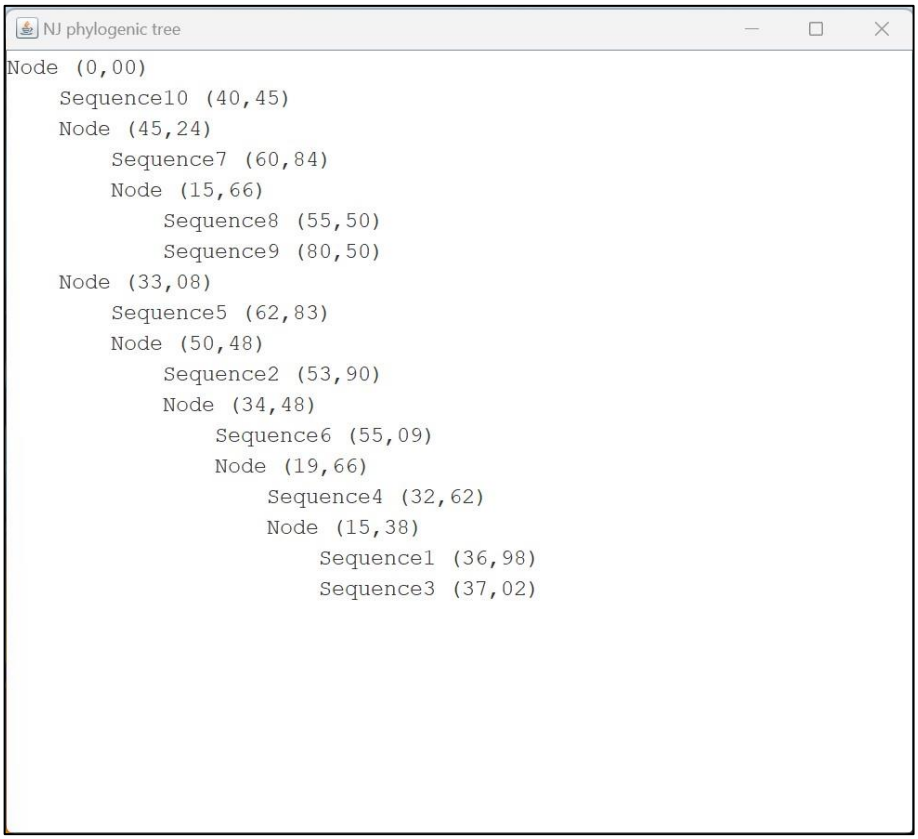
Annexe 15 : Arbre NJ pour la protéine Tat



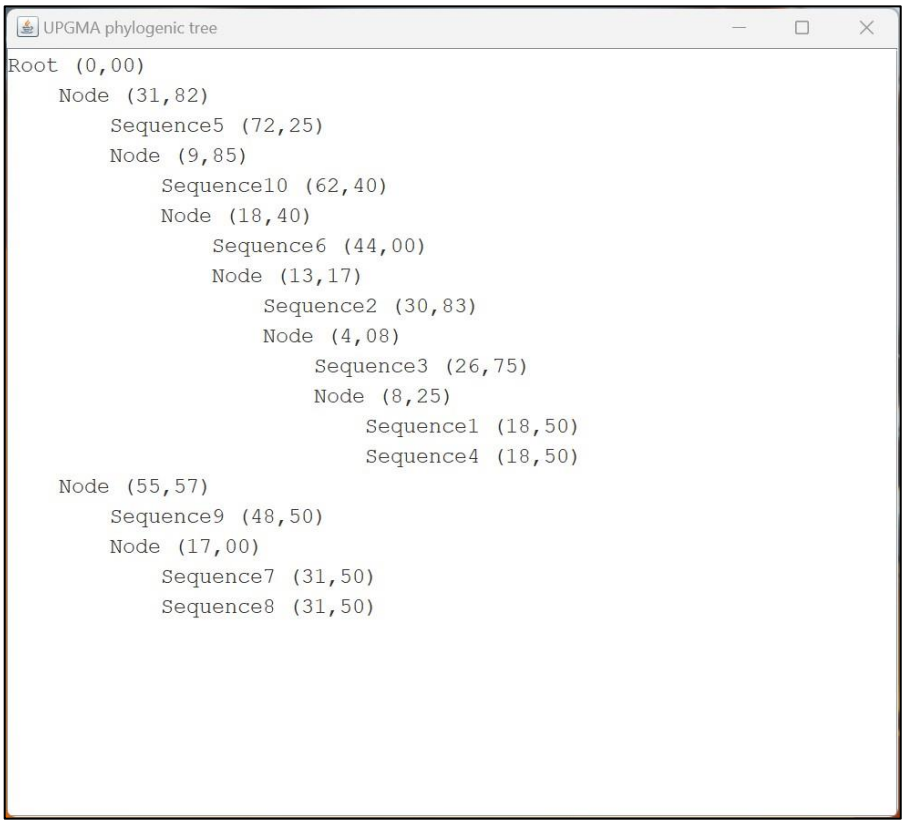
Annexe 16 : Arbre UPGMA pour les séquences ADN Env 300-600



Annexe 17 : Arbre NJ pour les séquences ADN Env 300-600



Annexe 18 : Arbre UPGMA pour les séquences ADN Env 600-900



Annexe 19 : Arbre NJ pour les séquences ADN Env 600-900