Evolution of repetitive sequences in viruses

CS516 Bioinformatics - Project 3

Israel Gonzalez S.

4/27/23

Table of contents

1 Abstract **2**

2 Introduction **2**

3 Methods **2**
3.1 Data structures and algorithms . . . 3
3.1.1 Task 1: Calculating pairwise Hamming distance . . . 3
3.1.2 Task 2: Building a rooted phylogenetic tree . . . 7
3.1.3 Task 3: Inferring ancestral sequences . . . 8
3.1.4 Task 4 and Task E1: The evolution of the most frequent 9-mer repeat in coronaviruses and in multiple aditional viruses . . . 12

4 Results **16**
4.1 Trees and sequences in Task 4 and Task E1 . . . 16
4.1.1 Brief description of viruses . . . 16
4.1.2 List of Genomes used . . . 16
4.1.3 Coronavirus-19 Results . . . 18
4.1.4 HIV-1 Results . . . 21
4.1.5 Adenovirus-2 Results . . . 23
4.1.6 Ebola Results . . . 25
4.1.7 Hepatitis-B Results . . . 28

5 Discussion **33**
5.1 Task 5 . . . 33
5.2 Task E2 . . . 34

6 Conclusion **36**

7 Distribution of work **36**

8 References **36**

1 Abstract

In this project, we study the evolution of the most frequent 9-mers in coronaviruses as main tasks and other viruses. We calculate sequence similarity, build a rooted phylogenetic tree, reconstruct ancestral sequences not only for the most frequent 9-mers as it was required but also for other k-mers that, in our opinion, could give more light to the genomic relations viruses represent.

2 Introduction

In our class, we have learned that analyzing the frequency of k-mers in genomes of living beings - like viruses - has the potential to provide relevant insights regarding the evolution and biological diversity of them. By calculating similarity or how distant their patterns are, building phylogennetic trees based on the most frequen k-mers (k=9 for this study), we can infer genomic relationships and if they possess common ancestors. In this sense, we can have a vision of the proximity the spread of viruses has produced on the virus types.

For instance, if a particular k-mer is found in many different viral genomes, it may indicate that this sequence has been conserved over time and is important for the virus's function or survival. On the other hand, if a k-mer is only found in a subset of viral genomes, it may indicate that this sequence has less in common and perhaps it has more biological reasons.

This project has five tasks centered on Coronavirus SARCoV-2 strains, and finally extends its methodology to two aditional taks related to Adenovirus-2, Ebola, HIV-1, and Hepatitis-B viruses; so that we, as students in the biotechnology field, can understand that the methodology is applicable to other genomic contexts also and learn to draw some conclusions based on the information we can retrieve from the data we analyze from complete genome sequences.

3 Methods

In this section, we proceed to describe the methods and data structures that we have implemented in Python to accomplish each required task of this project. We have organized in functions in files `task1.py`, `task2.py`, `task3.py`, and `task4_e1.py`

3.1 Data structures and algorithms

3.1.1 Task 1: Calculating pairwise Hamming distance

- Python Code

```
import numpy as np
from timeit import default_timer as timer

def hamming_distance(s1, s2):
  if len(s1) != len(s2):
    return None
  else:
    distance = 0
    for i in range(len(s1)):
      if s1[i] != s2[i]:
        distance += 1
    return distance

def triangular_inferior_format(matrix):
  triangular_matrix = []
  for i in range(len(matrix)):
    row = []
    for j in range(len(matrix)):
      if i >= j:
        row.append(matrix[i][j])
    triangular_matrix.append(row)
  return triangular_matrix

def get_sequences_from_fasta_file(file_path):
  sequences = []
  line_sequence = 1
  file = open(file_path, 'r')
  for line in file:
    if line_sequence%2 == 0:
      sequences.append(line.replace('\n', ''))
    line_sequence += 1
  file.close()
  return sequences

def hamming_distance_matrix(sequences, triangular=True, print_matrix=False):
  num_seqs = len(sequences)
```

```
distance_matrix = np.zeros((num_seqs, num_seqs), dtype=int)

start = timer()
for i in range(num_seqs):
  for j in range(num_seqs):
    if i != j:
      distance_matrix[i][j] = hamming_distance(sequences[i], sequences[j])
end = timer()
hamming_distance_matrix_time = end - start

distance_matrix_triangular = triangular_inferior_format(distance_matrix.tolist())

if print_matrix == True and triangular == False:
  print(distance_matrix)
if print_matrix == True and triangular == True:
  print(distance_matrix_triangular)

if triangular == True:
  return distance_matrix_triangular, hamming_distance_matrix_time
else:
  return distance_matrix, hamming_distance_matrix_time
```

The code includes three functions for calculating the Hamming distance between sequences, constructing a distance matrix, and converting the matrix to a triangular inferior format.

The hamming_distance function takes two sequences as input and returns the number of differences (or mismatches) between them. If the two sequences have different lengths, the function returns None. Otherwise, it counts the number of positions where the characters in the two sequences are different and returns the count.

The triangular_inferior_format function takes a matrix as input and returns a list of lists representing the lower triangular part of the matrix. It does this by iterating over the rows and columns of the input matrix and only adding values to the output list if the row index is greater than or equal to the column index. The resulting list of lists can be used to represent the distance matrix in a more compact format.

Te function get_sequences_from_fasta_file takes a path where a fasta file is with kmers sequences and retrive its sequences as a list.

The hamming_distance_matrix function takes a list of sequences as input and constructs a matrix of Hamming distances between all pairs of sequences. It first creates an empty matrix with dimensions equal to the number of sequences, and then fills in the matrix by calling the hamming_distance function for each pair of sequences. Since the matrix is symmetric, the function only calculates and stores the lower triangular part of the matrix to save space. Finally,

it returns the matrix in a triangular inferior format by calling the triangular_inferior_format function if it is required or just the squared matrix. Also, it returns its running time in seconds.

- Example of execution

```
if __name__ == '__main__':
  print()
  print("\nSequences from array")
  print("====================")

  # Directly specifying arrays and getting a normal squared matrix
  sequences = ["ACGTAGGCCT", "ATGTAAGACT", "TCGAGAGCAC", "TCGAAAGCAT"]
  print("Hamming distance matrix for sequences (normal format):\n",sequences,"\n")
  m, t = hamming_distance_matrix(sequences, False, True)
  print('\n  Runing time: {:.5f} seconds'.format(t))
  print()

  # Directly specifying arrays and getting an inferior triangular matrix as BioPython uses
  sequences = ["ACGT", "AGTT", "ATCC", "GTCA"]
  print("Hamming distance matrix for sequences (inf. triangular format):\n",sequences,"\n"
  m, t = hamming_distance_matrix(sequences, True, True)
  print('\n  Runing time: {:.5f} seconds'.format(t))
  print()

  print("\nSequences from Fasta files")
  print("==========================")

  # From fasta file, reads virus kmers and produces a normal squared matrix
  sequences = get_sequences_from_fasta_file('./fasta-sequences/fasta2.txt')
  print("Hamming distance matrix for sequences (normal format):\n",sequences,"\n")
  m, t = hamming_distance_matrix(sequences, False, True)
  print('\n  Runing time: {:.5f} seconds'.format(t))
  print()

  # From fasta file, reads virus kmers and produces an inferior triangular matrix as BioPy
  sequences = get_sequences_from_fasta_file('./fasta-sequences/fasta2.txt')
  print("Hamming distance matrix for sequences (inf. triangular format):\n",sequences,"\n"
  m, t = hamming_distance_matrix(sequences, True, True)
  print('\n  Runing time: {:.5f} seconds'.format(t))
  print()
```

- Output of Example of execution

```
PS C:\Development\nmsu\nmsu-cs516-bioinformatics-project\ps03> py .\task1.py


Sequences from array
====================
Hamming distance matrix for sequences (normal format):
 ['ACGTAGGCCT', 'ATGTAAGACT', 'TCGAGAGCAC', 'TCGAAAGCAT']

[[0 3 6 4]
 [3 0 7 5]
 [6 7 0 2]
 [4 5 2 0]]

  Runing time: 0.00002 seconds

Hamming distance matrix for sequences (inf. triangular format):
 ['ACGT', 'AGTT', 'ATCC', 'GTCA']

[[0], [2, 0], [3, 3, 0], [4, 4, 2, 0]]

  Runing time: 0.00002 seconds


Sequences from Fasta files
==========================
Hamming distance matrix for sequences (normal format):
 ['ACGTAGGCCT', 'ATGTAAGACT', 'TCGAGAGCAC', 'TCGAAAGCAT']

[[0 3 6 4]
 [3 0 7 5]
 [6 7 0 2]
 [4 5 2 0]]

  Runing time: 0.00002 seconds

Hamming distance matrix for sequences (inf. triangular format):
 ['ACGTAGGCCT', 'ATGTAAGACT', 'TCGAGAGCAC', 'TCGAAAGCAT']

[[0], [3, 0], [6, 7, 0], [4, 5, 2, 0]]

  Runing time: 0.00002 seconds
```

3.1.2 Task 2: Building a rooted phylogenetic tree

- Python Code

```
from Bio import Phylo
from Bio.Phylo.TreeConstruction import DistanceMatrix
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor
from task1 import hamming_distance_matrix, get_sequences_from_fasta_file
from timeit import default_timer as timer

def tree_by_neighbor_joining(sequences, plot = False, plot_type='ascii'):
  start = timer()
  distance_matrix, distance_matrix_time = hamming_distance_matrix(sequences)
  biopython_dist_matrix = DistanceMatrix(names=sequences, matrix=distance_matrix)
  constructor = DistanceTreeConstructor()
  tree = constructor.nj(biopython_dist_matrix)
  midpoint_clade = list(tree.find_clades(name="midpoint"))
  tree.root_with_outgroup(midpoint_clade)
  end = timer()
  tree_generation_time = end - start
  if plot and plot_type == 'ascii':
    Phylo.draw_ascii(tree)
  if plot and plot_type == 'image':
    Phylo.draw(tree)
  return tree, tree_generation_time
```

This code defines a function called tree_by_neighbor_joining that constructs a phylogenetic tree using the neighbor-joining method. It takes a list of sequences as input and can optionally plot the resulting tree.

The first step in the function is to compute the distance matrix using the hamming_distance_matrix function from task1. The distance matrix is then converted into a DistanceMatrix object from Biopython, which is needed for the neighbor-joining algorithm.

The next step is to construct the tree using the nj method from DistanceTreeConstructor, which implements the neighbor-joining algorithm. The midpoint clade is then found and used as the root of the tree.

If plot is set to True, the function plots the resulting tree using either ASCII art or an image, depending on the value of plot_type. Finally, the function returns the resulting tree object. . Also, it returns its running time in seconds.

- Example of execution

```
if __name__ == '__main__':
  print()
  #sequences = ["ACGTAGGCCT", "ATGTAAGACT", "TCGAGAGCAC", "TCGAAAGCAT"]
  sequences = get_sequences_from_fasta_file('./fasta-sequences/fasta2.txt')
  print("Rooted phylogenetic tree (Neighbor Joining) for sequences:\n\n",sequences,"\n")
  tree, tree_generation_time = tree_by_neighbor_joining(sequences, True, "ascii")
  print('Runing time: {:.5f} seconds'.format(tree_generation_time))
  print()
```

- Output of Example of execution

```
PS C:\Development\nmsu\nmsu-cs516-bioinformatics-project\ps03> py .\task2.py

Rooted phylogenetic tree (Neighbor Joining) for sequences:

 ['ACGTAGGCCT', 'ATGTAAGACT', 'TCGAGAGCAC', 'TCGAAAGCAT']

                                          ____________ ACGTAGGCCT
  ________________________________________|
 |                                        |____________________________ ATGTAAGACT
_|
 |_________________________ TCGAGAGCAC
 |
 | TCGAAAGCAT

Runing time: 0.00036 seconds
```

3.1.3 Task 3: Inferring ancestral sequences

- Python Code

```
import numpy as np
from timeit import default_timer as timer
from task2 import tree_by_neighbor_joining

nucleotides = 'ACGT'

def small_parsimony(tree):
  start = timer()
  sequences = [node.name for node in tree.get_terminals()]
```

```
  # (i-th) letter of each sequence
  words = []
  for i in range(0, len(sequences[0])):
    word = ''
    for sequence in sequences:
      word += str(sequence[i])
    words.append(word)

  ancestral_sequence = ''
  for word in words:
    ancestral_sequence += sp(word)
  end = timer()
  ancestral_sequence_time = end - start
  return ancestral_sequence, ancestral_sequence_time

def sp(word):
  left, right = [], []
  for i, c in enumerate(word):
    if i % 2 == 0:
      left.append(c)
    else:
      right.append(c)

  aditional = False
  if len(right) % 2 != 0:
    aditional = True

  scores_boths = []
  for i, c in enumerate(left):
    c_left = c
    if i == len(left) - 1 and aditional == True:
      c_right = None
    else:
      c_right = right[i]
    scores_left = [0 if c_left == n else float('inf') for n in nucleotides]
    if c_right != None:
      scores_right = [0 if c_right == n else float('inf') for n in nucleotides]
    else:
      scores_right = [float('inf') for n in nucleotides]
    scores_both = scores(scores_left, scores_right)
    scores_boths.append(scores_both)
```

```
  while len(scores_boths) > 1:
    num_pairs = len(scores_boths) // 2
    pairs = []
    for i in range(num_pairs):
      start_index = i * 2
      end_index = start_index + 2
      pair = scores_boths[start_index:end_index]
      pairs.append(pair)
    scoresb = []
    for pair in pairs:
      scoresb.append(scores(pair[0], pair[1]))
    scores_boths = scoresb

  scores_boths = np.array(scores_boths).flatten()

  nucleotide_letter = None
  for i, s in enumerate(scores_boths):
    if s == min(scores_boths):
      nucleotide_letter = nucleotides[i]
  return nucleotide_letter

def scores(scores_node_left, scores_node_right):
  scores_internal_node = []
  for ni, n in enumerate(nucleotides):
    ss_left = []
    for si,s in enumerate(scores_node_left):
      ss = s + (0 if si == ni else 1)
      ss_left.append(ss)
    ss_right = []
    for si,s in enumerate(scores_node_right):
      ss = s + (0 if si == ni else 1)
      ss_right.append(ss)
    score = min(ss_left) + min(ss_right)
    scores_internal_node.append(score)
  return scores_internal_node
```

The small_parsimony function takes a phylogenetic tree as input, extracts the DNA sequences of the terminal nodes, computes the ancestral sequence using the small parsimony algorithm (implemented in the sp function), and returns the ancestral sequence and the time it took to compute it. Specifically, the function creates a list of strings (sequences) from the names of the terminal nodes of the tree. It then loops over the letters of the first sequence in sequences, and creates a list (words) of strings where each string contains the i-th letter of all sequences.

Then, the sp function is then called on each word to compute the corresponding letter of the ancestral sequence. Finally, the ancestral sequence is built by concatenating all letters computed by the sp function. The small_parsimony function returns the ancestral sequence and the time it took to compute it in seconds.

The sp function takes a DNA word as input and computes the letter of the ancestral sequence that corresponds to the DNA position represented by that word using the small parsimony algorithm. The function first splits the input word into two lists (left and right) that contain the nucleotides of the first and second sequences, the second and third sequences, and so on. It then determines whether an additional nucleotide needs to be added to the right list if the length of the original sequence list was odd. The function initializes an empty list (scores_boths) that will store the scores of all possible nucleotides for each internal node in the tree. It then loops over the nucleotides in the left list, and computes the score of each nucleotide by comparing it to the nucleotide in the corresponding position in the right list, or to None if the right list is shorter by one element. The scores function is then called to compute the scores of the internal nodes in the tree. The function repeatedly applies the scores function to pairs of internal nodes until a single score is obtained, which is then converted into a nucleotide letter by finding the index of the minimum score in the scores_boths list. The function returns the nucleotide letter.

The scores function takes two lists of scores (scores_node_left and scores_node_right) as input, and returns a list of scores for the internal node in the tree. The function initializes an empty list (scores_internal_node) that will store the scores of all possible nucleotides for the internal node. It then loops over the nucleotides in the nucleotides list, and for each nucleotide, it computes the sum of the score of the nucleotide in the scores_node_left list and the score of the corresponding nucleotide in the scores_node_right list. The function returns the list of scores for the internal node.

- Example of execution

```
if __name__ == '__main__':
  print()
  sequences = ["ACGTAGGCCT", "ATGTAAGACT", "TCGAGAGCAC", "TCGAAAGCAT"]
  tree = tree_by_neighbor_joining(sequences, True, "ascii")
  ancestral_sequence, ancestral_sequence_time = small_parsimony(tree[0])
  print("Ancestral Sequence (Small Parsimony): ",ancestral_sequence,"\n")
  print('Runing time: {:.5f} seconds'.format(ancestral_sequence_time))
  print()
```

- Output of Example of execution

```
PS C:\Development\nmsu\nmsu-cs516-bioinformatics-project\ps03> py .\task3.py
```

```
                                              ____________ ACGTAGGCCT
  ____________________________________________|
 |                                            |_____________________________ ATGTAAGACT
_|
 |_____________________________ TCGAGAGCAC
 |
 | TCGAAAGCAT
```

```
Ancestral Sequence (Small Parsimony):  TCGTAAGCCT

Runing time: 0.00039 seconds
```

3.1.4 Task 4 and Task E1: The evolution of the most frequent 9-mer repeat in coronaviruses and in multiple aditional viruses

- Python Code

```
import os
import sys
import numpy as np
from collections import Counter
from FrequentKmers import FrequentKmers
from datasets import datasets
from task2 import tree_by_neighbor_joining

def get_genome_files(dir_path):
  res = []
  for path in os.listdir(dir_path):
    if os.path.isfile(os.path.join(dir_path, path)):
      res.append(path)
  return res

def get_most_frequent_nine_mer(nine_mers_dict):
  max_value = 0
  max_key = None
  for index,value in enumerate(nine_mers_dict):
    if nine_mers_dict[value] > max_value:
      max_value = nine_mers_dict[value]
      max_key = value
  return max_key, max_value
```

```
def get_kmers(dir_path, klength):
  fk = FrequentKmers()
  files = get_genome_files(dir_path)
  covid_k = [klength]

  nine_mers = {}
  lst = []
  for file in files:
    covid2_genome = fk.get_covid_genome(dir_path + '/' + file)
    for k in covid_k:
      result = fk.better_frequent_words(covid2_genome, k)
      rs = result[0]
      r_len = len(result[0])
      if r_len < 1000:
        for r in rs:
          nine_mers[r] = nine_mers[r] + 1 if r in nine_mers else 1
          virus = dir_path.split('/')[len(dir_path.split('/')) - 1]
          dataset_virus = datasets[virus][file.replace(".txt", "")]["title"]
          l = {}
          l["v"] = dataset_virus
          l["k"] = k
          l["s"] = r
          lst.append(l)

  kmer_list = [d['s'] for d in lst]
  counter = Counter(kmer_list)
  sorted_counts = sorted(counter.items(), key=lambda x: x[1], reverse=True)

  print("| Sequence | Count | Genome |")
  print("|:--------:|:-----:|:------:|")
  for s, count in sorted_counts:
    genome_titles = [d['v'] for d in lst if d['s'] == s]
    print(f"| {s} | {count} | {genome_titles} |")

  return nine_mers

def get_sequences(nine_mers):
  sequences = []
  for s in nine_mers.keys():
    sequences.append(s)
  return sequences
```

```
def generate_mutations(kmer):
  mutations = []
  nucleotides = ['A','C','T','G']
  for i in range(len(kmer)):
    for nucleotide in nucleotides:
      if kmer[i] != nucleotide:
        mutation = kmer[:i] + nucleotide + kmer[i+1:]
        if mutation not in mutations:
          mutations.append(mutation)
  return mutations
```

This is a script that uses the FrequentKmers class and the tree_by_neighbor_joining function to analyze genomes and generate a rooted phylogenetic tree. Here's an explanation of each function:

get_genome_files(dir_path): This function takes a directory path as input and returns a list of the filenames in that directory.

get_most_frequent_nine_mer(nine_mers_dict): This function takes a dictionary of 9-mers and their frequencies and returns the most frequent 9-mer and its frequency.

get_kmers(dir_path, klength): This function takes a directory path and a k-mer length as inputs, and returns a dictionary of 9-mers and their frequencies. It uses the FrequentKmers class to read in genome files from the specified directory, and finds the most frequent k-mers in each file using the better_frequent_words method. It prints a table with the filename, k-mer length, and sequence for each frequent k-mer found.

get_sequences(nine_mers): This function takes a dictionary of 9-mers and their frequencies and returns a list of the 9-mers.

- Example of execution

```
if __name__ == '__main__':

  klength = int(sys.argv[1]) if len(sys.argv) > 1 else 9
  type_drawing = "ascii"

  title = f"most frequent {klength}-mers"
  print()
  print("=========================================")
  print(f"Analysis of the {title}")
  print("=========================================")
  print()
```

```
    print(f"Coronavirus {title}")
    print("=========================================")
    nine_mers = get_kmers('./genomes/Coronavirus',klength)
    sequences = get_sequences(nine_mers)
    tree = tree_by_neighbor_joining(sequences, True, type_drawing)
    print('Runing time: {:.5f} seconds\n'.format(tree[1]))
    print()
```

- Output of Example of execution

```
PS C:\Development\nmsu\nmsu-cs516-bioinformatics-project\ps03> py .\task4_e1.py 9

=========================================
Analysis of the most frequent 9-mers
=========================================

Coronavirus most frequent 9-mers
=========================================
| Sequence | Count | Genome |
|:--------:|:-----:|:------:|
| TAAACGAAC | 3 | ['SARS coronavirus isolate CFB/SZ/94/03', 'Bat coronavirus isolate BANAL-2
| TTAACGAAC | 1 | ['Middle East respiratory syndrome coronavirus isolate MERS-CoV/THA/CU/17_
| TAACGAACT | 1 | ['Middle East respiratory syndrome coronavirus isolate MERS-CoV/THA/CU/17_
| TAATGGTAA | 1 | ['Pangolin coronavirus HKU4/P251T/pangolin/2018'] |

                                            , TAAACGAAC
  ------------------------------------------|
 |                                          |_____________________ TTAACGAAC
_|
 |_____________________________________________________________________ TAACGAACT
 |
 |_________________________________________ TAATGGTAA

Runing time: 0.00044 seconds
```

4 Results

4.1 Trees and sequences in Task 4 and Task E1

In this section, we present the results for obtained sequences, tree, and runing time for each virus studied:

4.1.1 Brief description of viruses

Our results involve the execution of the previous explained functions in these viruses:

- **Coronavirus-19**: Also known as SARS-CoV-2, it is a virus responsible for the COVID-19 pandemic. It belongs to the family Coronaviridae and is a single-stranded RNA virus that primarily infects the respiratory system in humans. The first cases of COVID-19 caused by the coronavirus-2 (SARS-CoV-2) were reported in Wuhan, China in December 2019.
- **HIV-1**: A human immunodeficiency virus that is responsible for acquired immunodeficiency syndrome (AIDS). HIV-1 is a retrovirus that primarily infects CD4+ T cells of the immune system, and if left untreated can lead to severe immune dysfunction.The first known case of HIV-1 infection was in a man from the Democratic Republic of Congo in 1959.
- **Adenovirus-2**: A non-enveloped, double-stranded DNA virus that infects humans and other animals. It belongs to the family Adenoviridae and is a common cause of respiratory and eye infections. Adenovirus-2 was first isolated from human adenoids in 1953.
- **Ebola**: A virus that causes severe hemorrhagic fever in humans and other primates. Ebola virus is a single-stranded RNA virus belonging to the family Filoviridae, and is spread through contact with bodily fluids of infected individuals. The first cases of Ebola virus disease (EVD) were reported in Sudan and the Democratic Republic of Congo in 1976.
- **Hepatitis-B**: A virus that primarily infects the liver and can cause acute and chronic hepatitis. Hepatitis B virus (HBV) is a partially double-stranded DNA virus belonging to the family Hepadnaviridae, and is spread through blood and bodily fluids. It can lead to serious liver complications such as cirrhosis and liver cancer. The hepatitis B virus was discovered in the serum of an Australian Aboriginal in 1965.

4.1.2 List of Genomes used

The are the variants by virus type with their Fasta code and description. It is important to note that we have used complete genomes in this study.

Table 1: Genomes used table

Virus Type	Fasta Genome	Title
Coronavirus	NC_045512.2	Coronavirus 2 isolate Wuhan-Hu-1
Coronavirus	KT225476.2	Middle East respiratory syndrome coronavirus isolate MERS-CoV/THA/CU/17_06_2015
Coronavirus	MZ937003.2	Bat coronavirus isolate BANAL-20-236/Laos/2020
Coronavirus	AY545919.1	SARS coronavirus isolate CFB/SZ/94/03
Coronavirus	OM009282.1	Pangolin coronavirus HKU4/P251T/pangolin/2018
Coronavirus	OK091006.1	Coronavirus 2 Delta variant/2021
Coronavirus	AY572034.1	Coronavirus 2 JP_Hiro97618 RNA
Coronavirus	OP183454.1	Coronavirus 2 isolate SARS-CoV-2/human/EGY/OMICRON-40/2022
Coronavirus	2182752773	Coronavirus 2 isolate SARS-CoV-2/human/CAN/Kappa_NML_P4/2021
Coronavirus	MZ496616.1	Coronavirus 2 isolate SARS-CoV-2/Felis catus/PER/UPCH_sc2_cat3/2021
Coronavirus	OQ780237.1	ORFB1ab USA human/USA/WA-PHL-027795/2022
Coronavirus	OQ779361.1	SARS-CoV-2/human/USA/WA-PHL-026151/2022
Coronavirus	OQ258758.1	Human/USA/NC-CDC-LC0984552/2023
Coronavirus	OQ722344.1	SARS-CoV-2/human/MAR/Omicron BA.2/2022
Coronavirus	OQ525773.1	SARS-CoV-2/human/BRA/ILMD2203877/2022
Coronavirus	OX460957.1	Covid Switzerland 2023
Coronavirus	LC594651.1	Coronavirus 2 JP_Hiro97618 RNA
HIV-1	MN692147.1	HIV-1 isolate Pt1-M26-NFL-30-PBio from USA
HIV-1	MH327766.1	HIV-1 isolate 1031 from Philippines
HIV-1	EU541617.1	HIV-1 clone pIIIB from USA
HIV-1	AF224507.1	HIV-1 strain HIV-1wk from South Korea
HIV-1	KX232629.1	HIV-1 isolate PK040 from Pakistan
HIV-1	MG365771.1	HIV-1 isolate 01BRRJUD508 from Brazil
HIV-1	KY498771.1	HIV-1 isolate S4858 from Cameroon
Adenovirus-2	AC_000007.1	Human adenovirus 2
Adenovirus-2	KP279747.1	Unidentified adenovirus isolate GPAdV_5
Adenovirus-2	NC_027708.1	Skunk adenovirus PB1
Adenovirus-2	X73487.1	Adenovirus type 12 DNA
Adenovirus-2	MZ073342.1	Porcupine adenovirus isolate X-24879-18
Adenovirus-2	MZ073341.1	Raccoon adenovirus isolate W-0155-19B
Ebola	KU143834.1	Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9
Ebola	AF086833.2	Ebola virus - Mayinga, Zaire, 1976
Ebola	KU182904.1	Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626

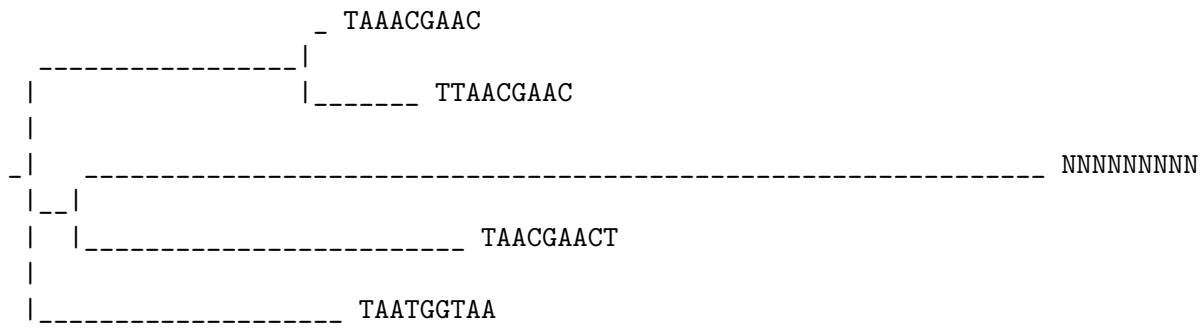
Virus Type	Fasta Genome	Title
Ebola	KM655246.1	Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran
Hepatitis-B	LC603637.1	Hepatitis B virus HB20-0698 DNA
Hepatitis-B	K01834.1	Duck hepatitis B virus
Hepatitis-B	AJ344115.1	Hepatitis B virus complete genome, genotype A
Hepatitis-B	KR014099.1	Hepatitis B virus strain #1958.7

4.1.3 Coronavirus-19 Results

4.1.3.1 Most frequent 9-mers:

The following are the results most frequent 9-mers for virus type: Coronavirus

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 9-mers**:

Table 2: Most frequent 9-mers Coronavirus table

Sequence	Count	Genome
TAAACGAAC	7	['SARS coronavirus isolate CFB/SZ/94/03', 'Coronavirus 2 JP_Hiro97618 RNA', 'Coronavirus 2 JP_Hiro97618 RNA', 'Coronavirus 2 isolate SARS-CoV-2/Felis catus/PER/UPCH_sc2_cat3/2021', 'Bat coronavirus isolate BANAL-20-236/Laos/2020', 'Coronavirus 2 isolate Wuhan-Hu-1', 'Coronavirus 2 isolate SARS-CoV-2/human/EGY/OMICRON-40/2022']
NNNNNNNNN	2	['Coronavirus 2 Delta variant/2021', 'SARS-CoV-2/human/USA/WA-PHL-026151/2022']
TTAACGAAC	1	['Middle East respiratory syndrome coronavirus isolate MERS-CoV/THA/CU/17_06_2015']

Sequence	Count	Genome
TAACGAACT	1	['Middle East respiratory syndrome coronavirus isolate MERS-CoV/THA/CU/17_06_2015']
TAATGGTAT	1	['Pangolin coronavirus HKU4/P251T/pangolin/2018']

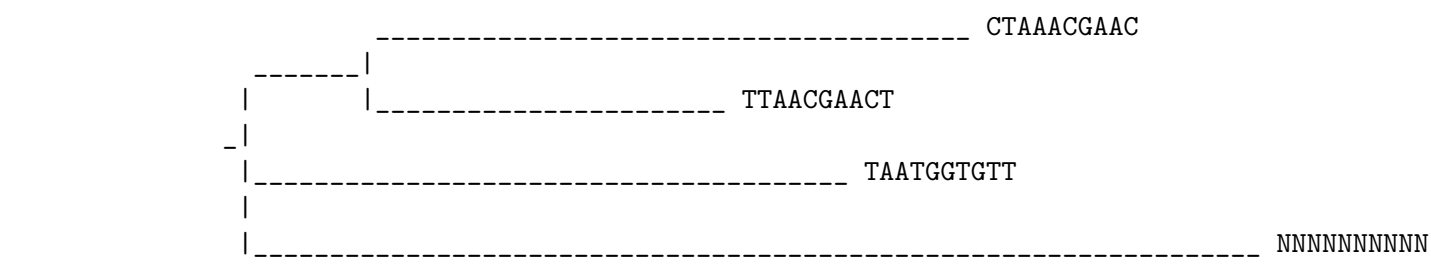
- **Running time**:

```
Runing time: 0.00054 seconds
```

4.1.3.2 Most frequent 10-mers:

The following are the results most frequent 10-mers for virus type: Coronavirus

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 10-mers**:

Table 3: Most frequent 10-mers Coronavirus table

Sequence	Count	Genome
CTAAACGAAC	7	['SARS coronavirus isolate CFB/SZ/94/03', 'Coronavirus 2 JP_Hiro97618 RNA', 'Coronavirus 2 JP_Hiro97618 RNA', 'Coronavirus 2 isolate SARS-CoV-2/Felis catus/PER/UPCH_sc2_cat3/2021', 'Bat coronavirus isolate BANAL-20-236/Laos/2020', 'Coronavirus 2 isolate Wuhan-Hu-1', 'Coronavirus 2 isolate SARS-CoV-2/human/EGY/OMICRON-40/2022']
TAATGGTGTT	3	['Coronavirus 2 JP_Hiro97618 RNA', 'Coronavirus 2 isolate SARS-CoV-2/Felis catus/PER/UPCH_sc2_cat3/2021', 'Coronavirus 2 isolate Wuhan-Hu-1']
TTAACGAACT	2	['Middle East respiratory syndrome coronavirus isolate MERS-CoV/THA/CU/17_06_2015', 'Pangolin coronavirus HKU4/P251T/pangolin/2018']

Sequence	Count	Genome
NNNNNNNNNNNNNNNNNN	2	[‘Coronavirus 2 Delta variant/2021’, ‘SARS-CoV-2/human/USA/WA-PHL-026151/2022’]

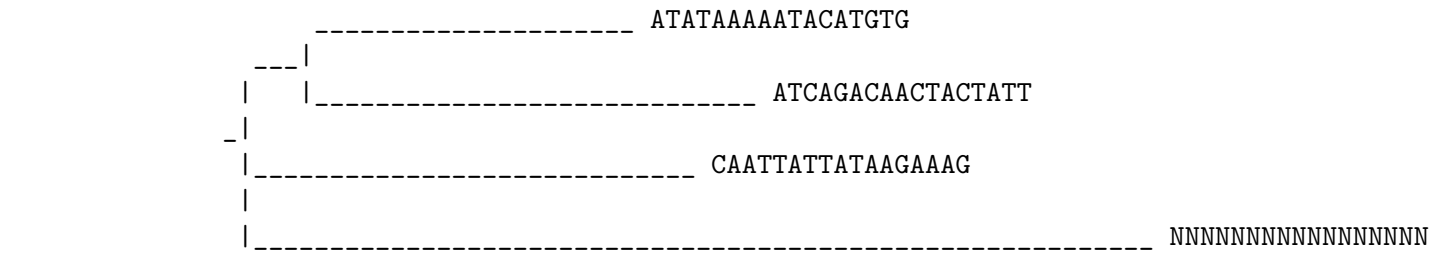
- **Running time**:

```
Runing time: 0.00054 seconds
```

4.1.3.3 Most frequent 17-mers:

The following are the results most frequent 17-mers for virus type: Coronavirus

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 17-mers**:

Table 4: Most frequent 17-mers Coronavirus table

Sequence	Count	Genome
CAATTATTATAAGAAAG	6	[‘Coronavirus 2 JP_Hiro97618 RNA’, ‘Coronavirus 2 isolate SARS-CoV-2/Felis catus/PER/UPCH_sc2_cat3/2021’, ‘Bat coronavirus isolate BANAL-20-236/Laos/2020’, ‘Coronavirus 2 isolate Wuhan-Hu-1’, ‘Coronavirus 2 isolate SARS-CoV-2/human/EGY/OMICRON-40/2022’, ‘SARS-CoV-2/human/USA/WA-PHL-026151/2022’]
ATCAGACAACTACTATT	5	[‘Coronavirus 2 JP_Hiro97618 RNA’, ‘Coronavirus 2 isolate SARS-CoV-2/Felis catus/PER/UPCH_sc2_cat3/2021’, ‘Coronavirus 2 isolate Wuhan-Hu-1’, ‘Coronavirus 2 isolate SARS-CoV-2/human/EGY/OMICRON-40/2022’, ‘SARS-CoV-2/human/USA/WA-PHL-026151/2022’]
NNNNNNNNNNNNNNNNNN	2	[‘Coronavirus 2 Delta variant/2021’, ‘SARS-CoV-2/human/USA/WA-PHL-026151/2022’]
ATATAAAAATACATGTG	1	[‘Bat coronavirus isolate BANAL-20-236/Laos/2020’]

- **Running time**:

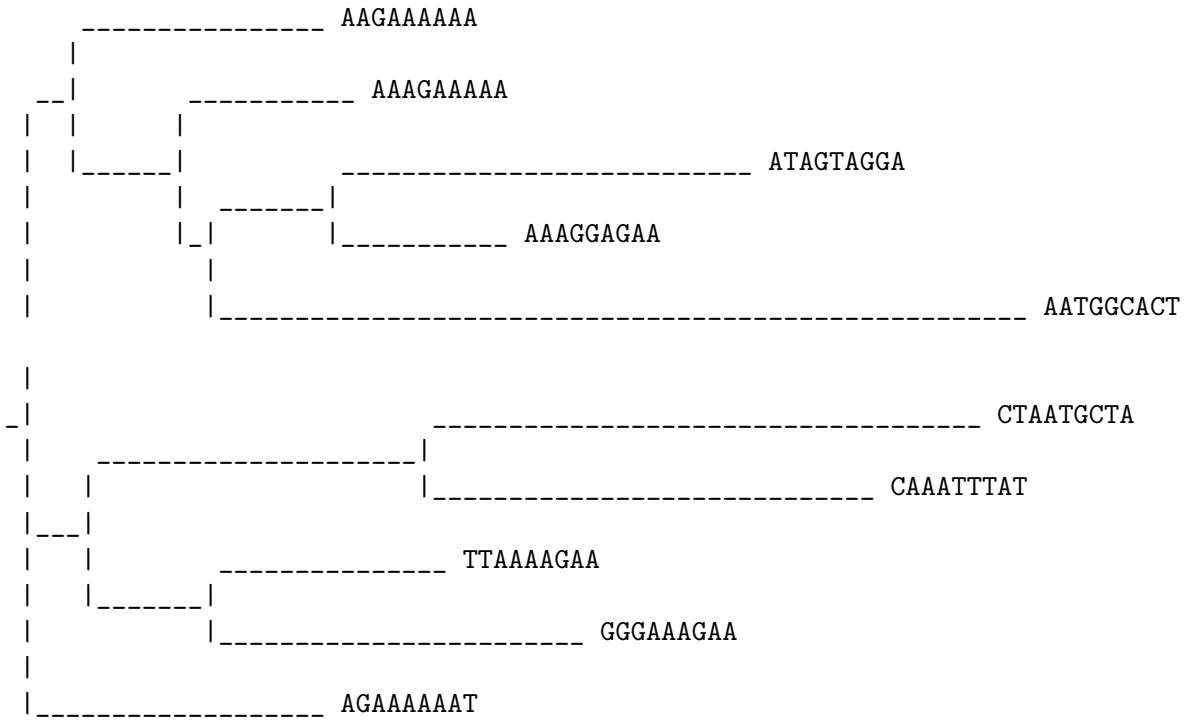
```
Runing time: 0.00077 seconds
```

4.1.4 HIV-1 Results

4.1.4.1 Most frequent 9-mers:

The following are the results most frequent 9-mers for virus type: HIV-1

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 9-mers**:

Table 5: Most frequent 9-mers HIV-1 table

Sequence	Count	Genome
AAGAAAAAA	3	['HIV-1 strain HIV-1wk from South Korea', 'HIV-1 clone pIIIB from USA', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA']
AAAGAAAAA	2	['HIV-1 isolate 01BRRJUD508 from Brazil', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA']

Sequence	Count	Genome
TTAAAAGA	2	['HIV-1 isolate 1031 from Philippines', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA']
AATGGCAC	1	['HIV-1 isolate PK040 from Pakistan']
GGGAAAGA	1	['HIV-1 isolate S4858 from Cameroon']
AAAGGAGA	1	['HIV-1 isolate S4858 from Cameroon']
AGAAAAAA	1	['HIV-1 isolate 1031 from Philippines']
CAAATTTA	1	['HIV-1 isolate 1031 from Philippines']
ATAGTAGG	1	['HIV-1 isolate 1031 from Philippines']
CTAATGCT	1	['HIV-1 isolate 1031 from Philippines']

- **Running time**:

```
Runing time: 0.00150 seconds
```

4.1.4.2 Most frequent 7-mers:

The following are the results most frequent 7-mers for virus type: HIV-1

- **Pylogenetic Tree by Neighbor Joining**:

- **Most frequent 7-mers**:

Table 6: Most frequent 7-mers HIV-1 table

Sequence	Count	Genome
AAGAAAA	4	['HIV-1 strain HIV-1wk from South Korea', 'HIV-1 isolate 01BRRJUD508 from Brazil', 'HIV-1 isolate 1031 from Philippines', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA']
AAAGAAA	2	['HIV-1 isolate PK040 from Pakistan', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA']
AAAAGAA	2	['HIV-1 isolate 01BRRJUD508 from Brazil', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA']
AGAAGAA	1	['HIV-1 clone pIIIB from USA']
AGCAGAA	1	['HIV-1 isolate S4858 from Cameroon']
AAAATTA	1	['HIV-1 isolate 1031 from Philippines']
AGAAAAA	1	['HIV-1 isolate 1031 from Philippines']

- **Running time**:

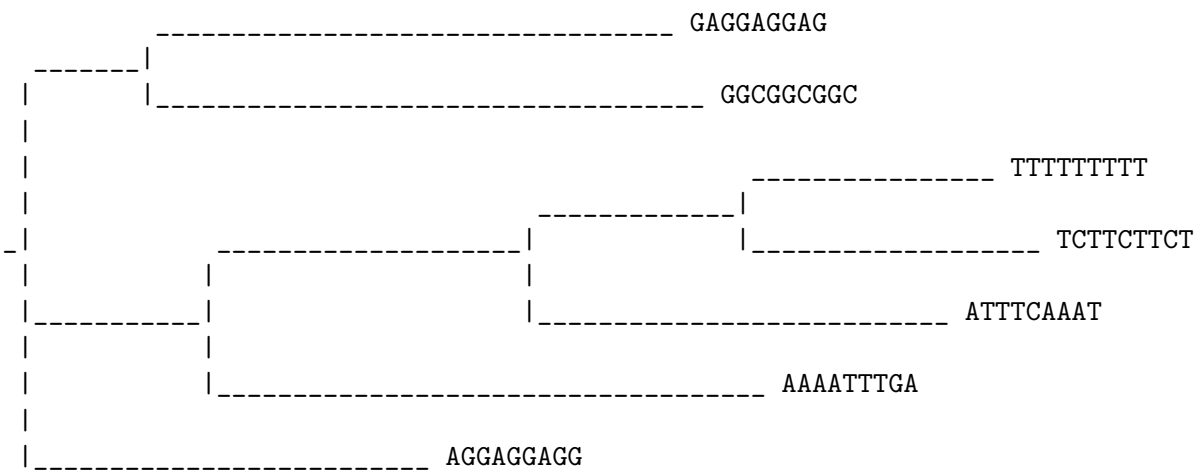
```
Runing time: 0.00098 seconds
```

4.1.5 Adenovirus-2 Results

4.1.5.1 Most frequent 9-mers:

The following are the results most frequent 9-mers for virus type: Adenovirus-2

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 9-mers**:

Table 7: Most frequent 9-mers Adenovirus-2 table

Sequence	Count	Genome
GAGGAGGA	38	['Raccoon adenovirus isolate W-0155-19B', 'Porcupine adenovirus isolate X-24879-18', 'Skunk adenovirus PB1']
GGCGGCGG	14	['Human adenovirus 2']
AAAATTTG	11	['Unidentified adenovirus isolate GPAdV_5']
ATTTCAAA	11	['Unidentified adenovirus isolate GPAdV_5']
TCTTCTTC	11	['Unidentified adenovirus isolate GPAdV_5']
TTTTTTTT	11	['Adenovirus type 12 DNA']
AGGAGGAG	10	['Adenovirus type 12 DNA']

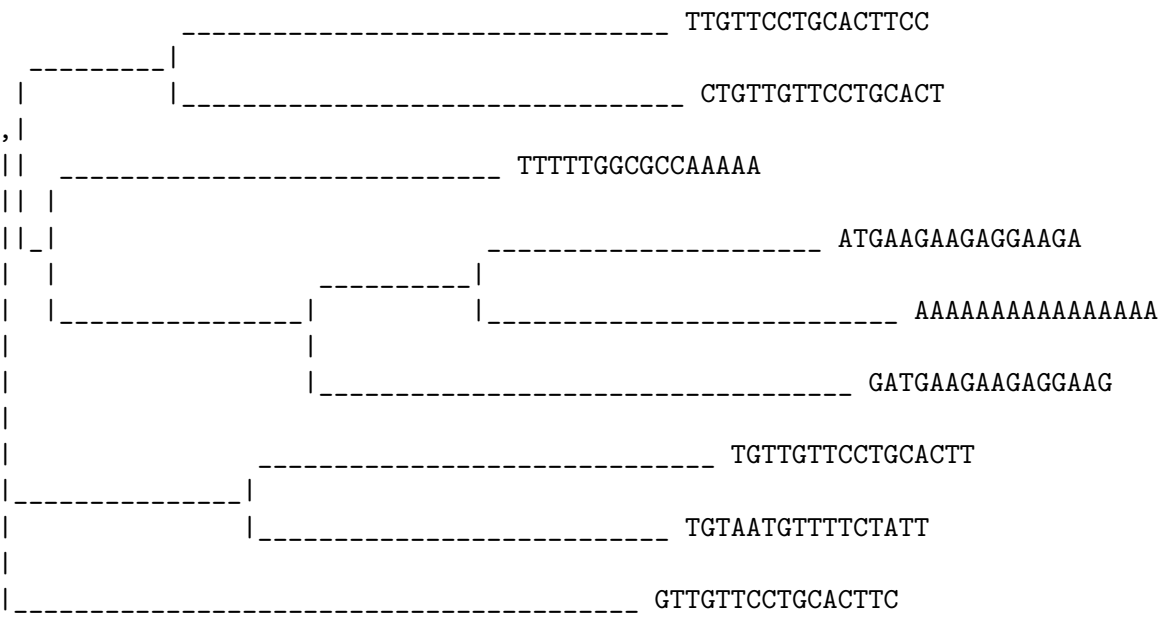
- **Running time**:

```
Runing time: 0.00073 seconds
```

4.1.5.2 Most frequent 16-mers:

The following are the results most frequent 16-mers for virus type: Adenovirus-2

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 16-mers**:

Table 8: Most frequent 16-mers Adenovirus-2 table

Sequence	Count	Genome
TTTTTGGC	3	[‘Raccoon adenovirus isolate W-0155-19B’, ‘Porcupine adenovirus isolate X-24879-18’, ‘Skunk adenovirus PB1’]
AAAAAAAA	1	[‘Human adenovirus 2’]
GATGAAGA	1	[‘Human adenovirus 2’]
ATGAAGAA	1	[‘Human adenovirus 2’]
TGTAATGT	1	[‘Unidentified adenovirus isolate GPAdV_5’]
CTGTTGTT	1	[‘Unidentified adenovirus isolate GPAdV_5’]
TGTTGTTC	1	[‘Unidentified adenovirus isolate GPAdV_5’]
GTTGTTCC	1	[‘Unidentified adenovirus isolate GPAdV_5’]
TTGTTCCT	1	[‘Unidentified adenovirus isolate GPAdV_5’]

- **Running time**:

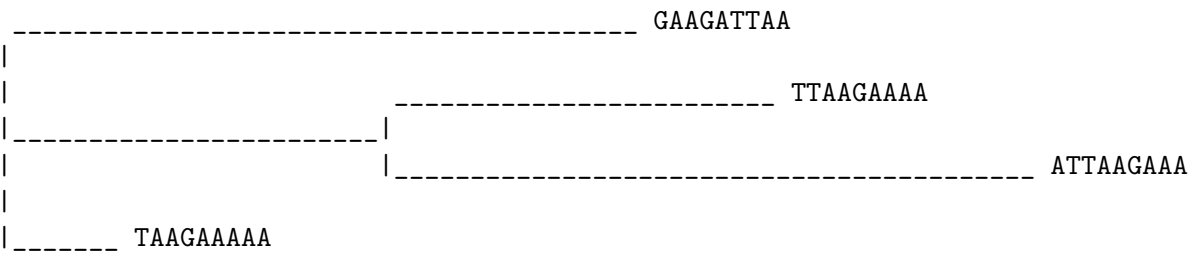
```
Runing time: 0.00132 seconds
```

4.1.6 Ebola Results

4.1.6.1 Most frequent 9-mers:

The following are the results most frequent 9-mers for virus type: Ebola

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 9-mers**:

Table 9: Most frequent 9-mers Ebola table

Sequence	Count	Genome
GAAGATTAA	4	['Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626']
TTAAGAAAA	4	['Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626']
TAAGAAAAA	4	['Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626']
ATTAAGAAA	3	['Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626']

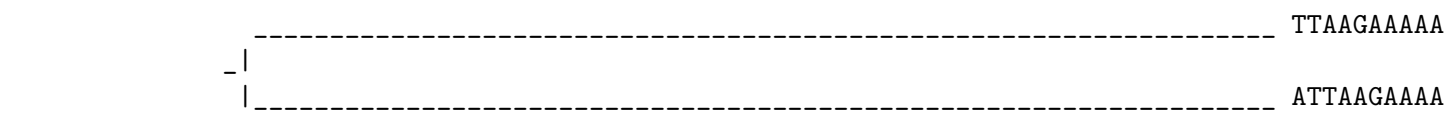
- **Running time**:

```
Runing time: 0.00032 seconds
```

4.1.6.2 Most frequent 10-mers:

The following are the results most frequent 10-mers for virus type: Ebola

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 10-mers**:

Table 10: Most frequent 10-mers Ebola table

Sequence	Count	Genome
TTAAGAAAAA	4	['Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626']
ATTAAGAAAA	3	['Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626']

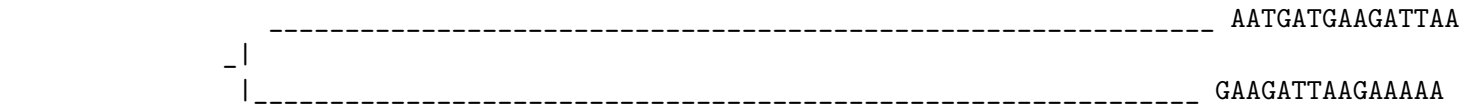
- **Running time**:

```
Runing time: 0.00037 seconds
```

4.1.6.3 Most frequent 15-mers:

The following are the results most frequent 15-mers for virus type: Ebola

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 15-mers**:

Table 11: Most frequent 15-mers Ebola table

Sequence	Count	Genome
GAAGATTAAGAAAAA	4	['Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626']
AATGATGAAGATTAA	1	['Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9']

- **Running time**:

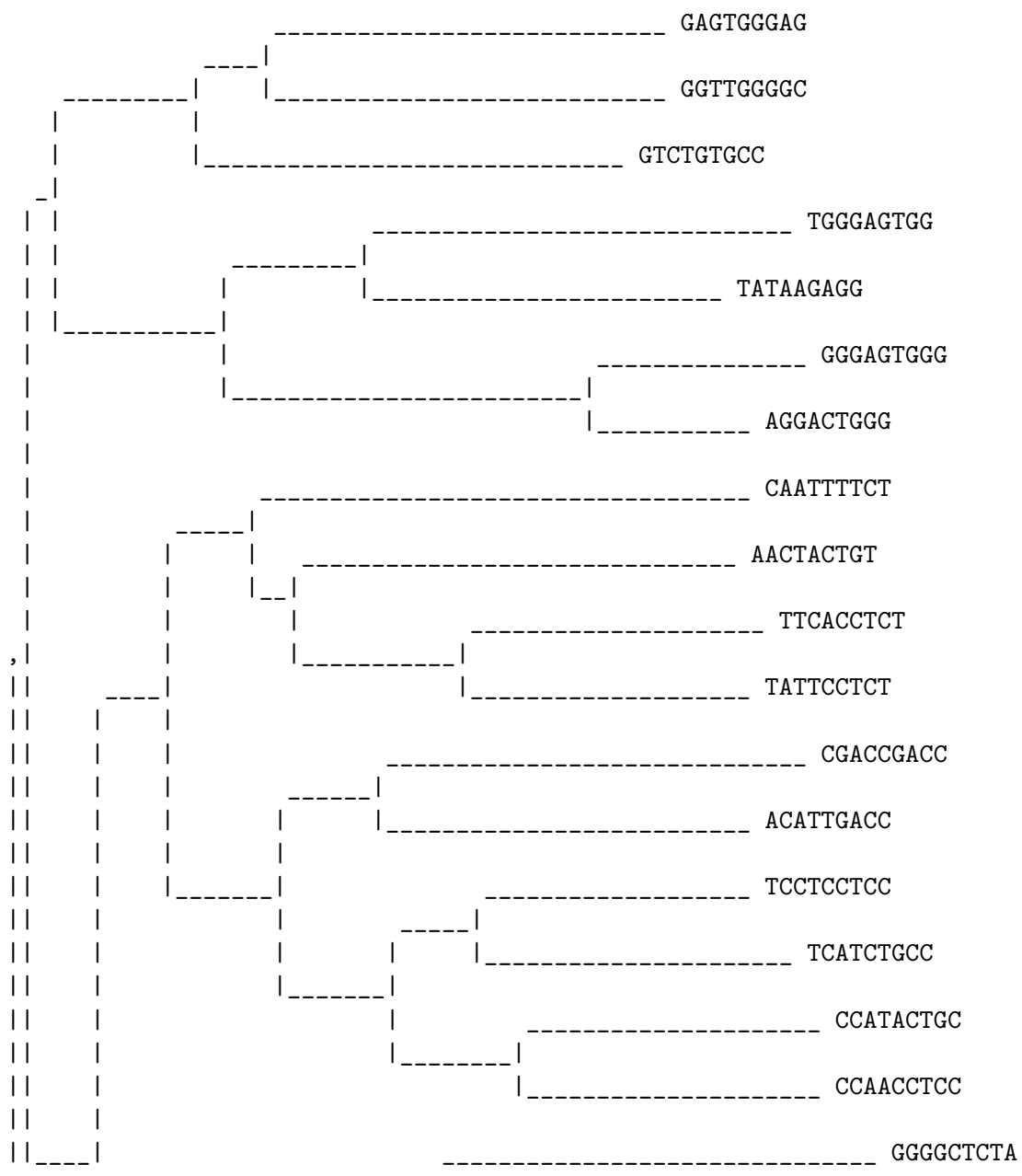
```
Runing time: 0.00037 seconds
```

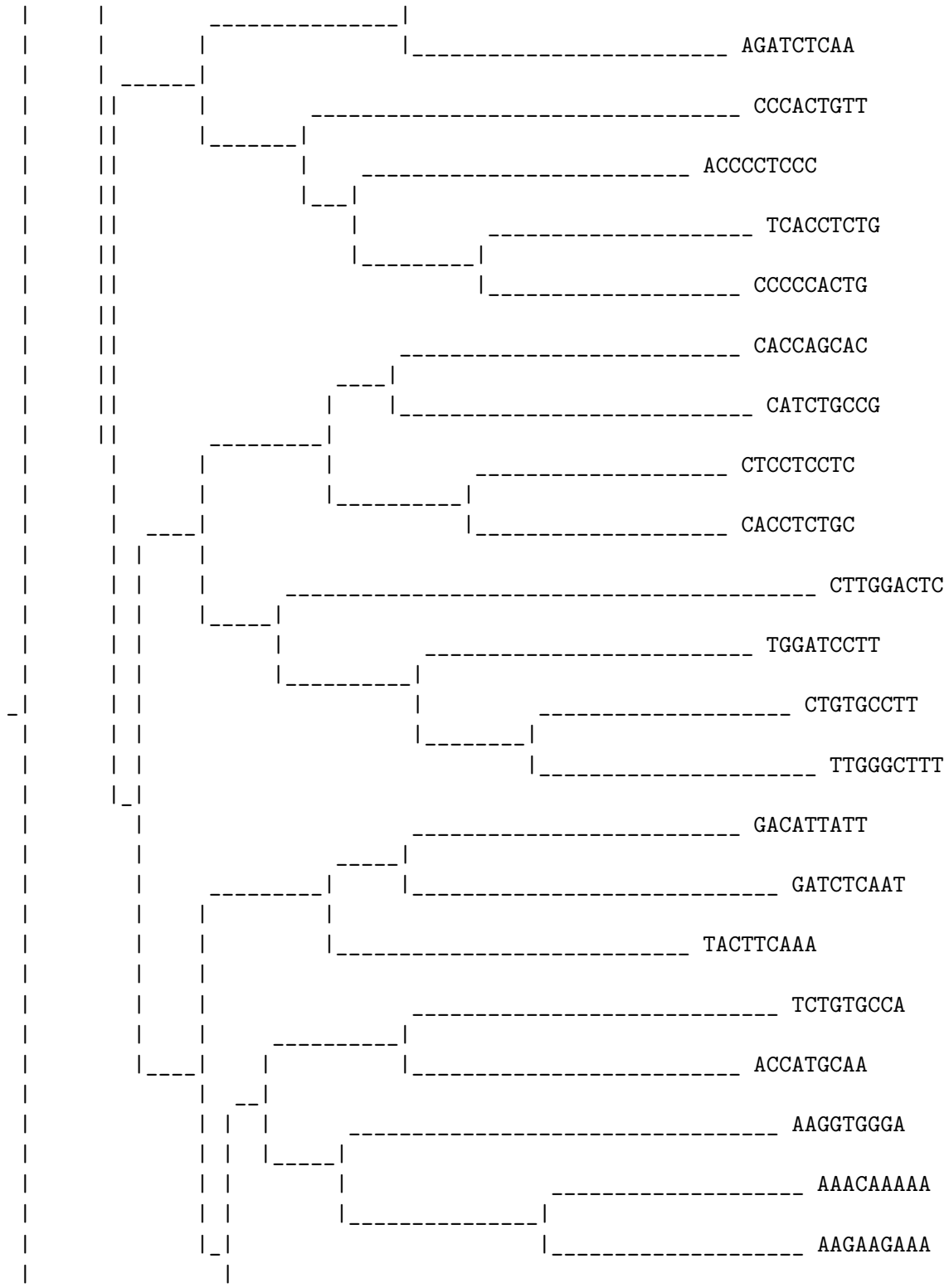
4.1.7 Hepatitis-B Results

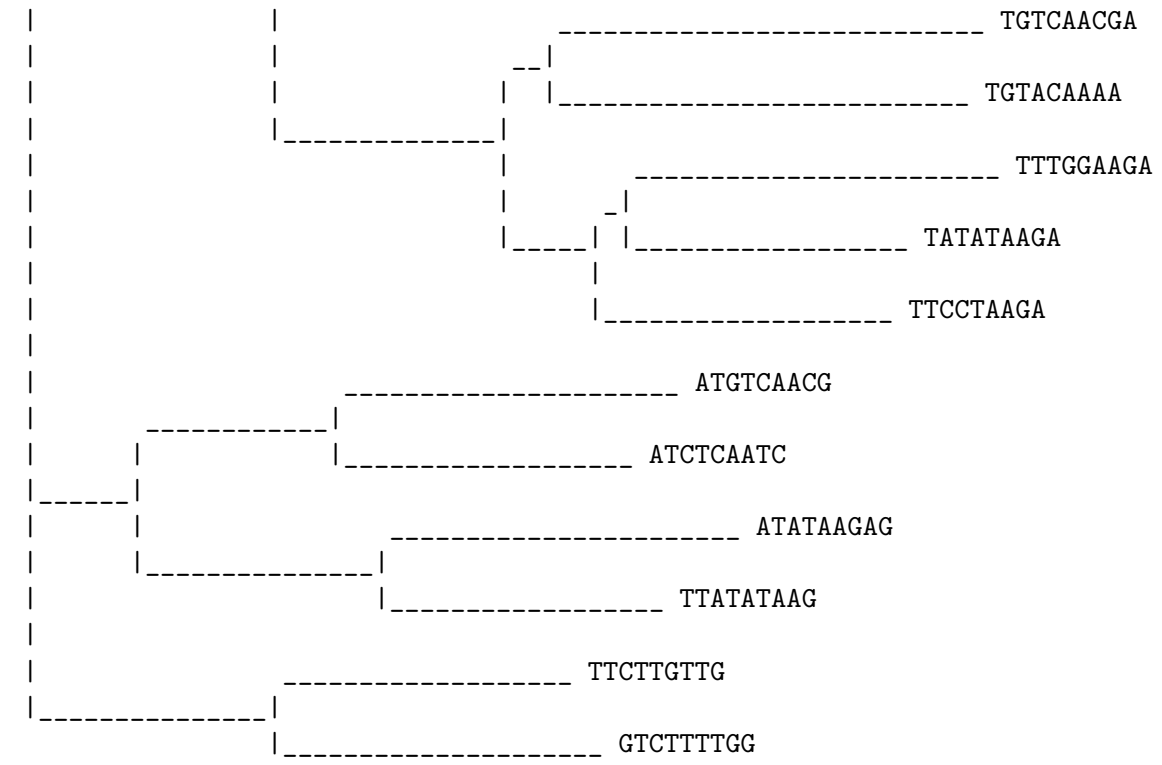
4.1.7.1 Most frequent 9-mers:

The following are the results most frequent 9-mers for virus type: Hepatitis-B

- **Pylogenetic Tree by Neighbor Joining**:




AGATCTCAA
CCCACTGTT
ACCCCTCCC
TCACCTCTG
CCCCCACTG
CACCAGCAC
CATCTGCCG
CTCCTCCTC
CACCTCTGC
CTTGGACTC
TGGATCCTT
CTGTGCCTT
TTGGGCTTT
GACATTATT
GATCTCAAT
TACTTCAAA
TCTGTGCCA
ACCATGCAA
AAGGTGGGA
AAACAAAAA
AAGAAGAAA



- **Most frequent 9-mers**:

Table 12: Most frequent 9-mers Hepatitis-B table

Sequence	Count	Genome
CCCCCACTG	8	['Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA']
CTGTGCCTT	3	['Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA']
TTCACCTCT	3	['Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA']
TCACCTCTG	3	['Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA']
CACCTCTGC	3	['Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA']
GTCTGTGCC	2	['Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus HB20-0698 DNA']
CTTGGACTC	2	['Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus HB20-0698 DNA']
GTCTTTTGG	2	['Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA']

Sequence	Count	Genome
AGGACTGGCG	1	['Hepatitis B virus complete genome, genotype A']
TTCCTAAGTA	1	['Hepatitis B virus complete genome, genotype A']
CCAACCTCTC	1	['Hepatitis B virus complete genome, genotype A']
TATTCCTCTT	1	['Hepatitis B virus complete genome, genotype A']
TGTACAAATA	1	['Hepatitis B virus complete genome, genotype A']
TTGGGCTTTT	1	['Hepatitis B virus complete genome, genotype A']
CCCACTGTTT	1	['Hepatitis B virus complete genome, genotype A']
TGGATCCTTT	1	['Hepatitis B virus complete genome, genotype A']
CCATACTGCC	1	['Hepatitis B virus complete genome, genotype A']
TTATATAAGT	1	['Hepatitis B virus complete genome, genotype A']
TATATAAGAA	1	['Hepatitis B virus complete genome, genotype A']
ATATAAGAGT	1	['Hepatitis B virus complete genome, genotype A']
TATAAGAGGG	1	['Hepatitis B virus complete genome, genotype A']
TACTTCAAAA	1	['Hepatitis B virus complete genome, genotype A']
ACATTGACCC	1	['Hepatitis B virus complete genome, genotype A']
AGATCTCAAA	1	['Hepatitis B virus complete genome, genotype A']
GATCTCAATT	1	['Hepatitis B virus complete genome, genotype A']
ATCTCAATCC	1	['Hepatitis B virus complete genome, genotype A']
GACATTATTT	1	['Hepatitis B virus complete genome, genotype A']
AAGAAGAAAA	1	['Duck hepatitis B virus']
ACCATGCAAA	1	['Hepatitis B virus strain #1958.7']
TCATCTGCCC	1	['Hepatitis B virus strain #1958.7']
CATCTGCCCG	1	['Hepatitis B virus strain #1958.7']
TTTGGAAGAA	1	['Hepatitis B virus strain #1958.7']
AACTACTGTT	1	['Hepatitis B virus strain #1958.7']
TTCTTGTTTG	1	['Hepatitis B virus HB20-0698 DNA']
CAATTTTCTT	1	['Hepatitis B virus HB20-0698 DNA']
CACCAGCACC	1	['Hepatitis B virus HB20-0698 DNA']
TGGGAGTGGG	1	['Hepatitis B virus HB20-0698 DNA']
GGGAGTGGGG	1	['Hepatitis B virus HB20-0698 DNA']
AAACAAAAAA	1	['Hepatitis B virus HB20-0698 DNA']
GGTTGGGGGC	1	['Hepatitis B virus HB20-0698 DNA']
ATGTCAACGG	1	['Hepatitis B virus HB20-0698 DNA']
TGTCAACGAA	1	['Hepatitis B virus HB20-0698 DNA']
TCTGTGCCAA	1	['Hepatitis B virus HB20-0698 DNA']
ACCCCTCCCC	1	['Hepatitis B virus HB20-0698 DNA']
GGGGCTCTAA	1	['Hepatitis B virus HB20-0698 DNA']
CGACCGACCC	1	['Hepatitis B virus HB20-0698 DNA']
GAGTGGGAAG	1	['Hepatitis B virus HB20-0698 DNA']
CTCCTCCTCC	1	['Hepatitis B virus HB20-0698 DNA']
TCCTCCTCCC	1	['Hepatitis B virus HB20-0698 DNA']

Sequence	Count	Genome
AAGGTGGGTA	1	[‘Hepatitis B virus HB20-0698 DNA’]

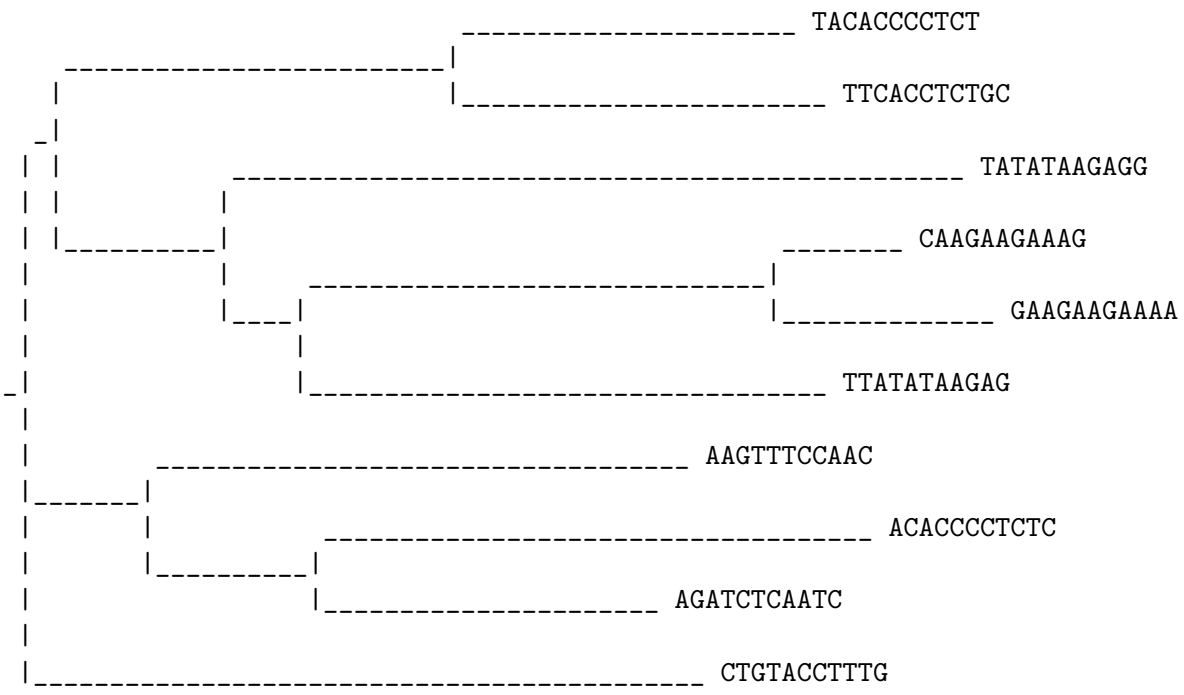
- **Running time**:

```
Runing time: 0.08194 seconds
```

4.1.7.2 Most frequent 11-mers:

The following are the results most frequent 11-mers for virus type: Hepatitis-B

- **Pylogenetic Tree by Neighbor Joining**:



- **Most frequent 11-mers**:

Table 13: Most frequent 11-mers Hepatitis-B table

Sequence	Count	Genome
TTCACCTCTTGC	3	[‘Hepatitis B virus complete genome, genotype A’, ‘Hepatitis B virus strain #1958.7’, ‘Hepatitis B virus HB20-0698 DNA’]
TTATATAAGAG	1	[‘Hepatitis B virus complete genome, genotype A’]

Sequence	Count	Genome
TATATAAGA	1	AGG ['Hepatitis B virus complete genome, genotype A']
AGATCTCA	1	ATC ['Hepatitis B virus complete genome, genotype A']
GAAGAAGA	1	AAA ['Duck hepatitis B virus']
AAGTTTCC	1	AAC ['Duck hepatitis B virus']
CAAGAAGA	1	AAG ['Duck hepatitis B virus']
CTGTACCT	1	TTG ['Duck hepatitis B virus']
TACACCCC	1	TCT ['Duck hepatitis B virus']
ACACCCCT	1	CTC ['Duck hepatitis B virus']

- **Running time**:

```
Runing time: 0.00200 seconds
```

5 Discussion

5.1 Task 5

The given results show the most frequent k-mers for different values of k (k=9,10,17) for the coronavirus genome. These values have been selected trying to produce optimal clustering among the studied genomes. The tables show the sequence, count, and genome where the k-mer was found. The pylogenetic tree by neighbor joining shows the relationships between the different sequences based on their similarities. The running time of the analysis is also reported. Also, the pylogenetic tree shows the relationships between these genome sequences based on their similarity. The running time of the analysis indicates the computational efficiency of the algorithm used to generate these results is directly proportional to the number of sequences involved. Overall, these results provide insights into the genomic diversity of coronavirus and how different sequences are related to each other. These findings could be used for further research on the evolution and transmission of coronavirus.

In case of Coronavirus, with k=9 we see a strong relation through 9-mer `TAAACGAAC` for 'SARS coronavirus isolate CFB/SZ/94/03', 'Coronavirus 2 JP_Hiro97618 RNA', 'Coronavirus 2 JP_Hiro97618 RNA', 'Coronavirus 2 isolate SARS-CoV-2/Felis catus/PER/UPCH_sc2_cat3/2021', 'Bat coronavirus isolate BANAL-20-236/Laos/2020', 'Coronavirus 2 isolate Wuhan-Hu-1', 'Coronavirus 2 isolate SARS-CoV-2/human/EGY/OMICRON-40/2022' and this cluster with a proximity in similarity with the 9-mer `TTAACGAAC` with the 'Coronavirus 2 Delta variant/2021', 'SARS-CoV-2/human/USA/WA-PHL-026151/2022' and the 'Middle East respiratory' variants.

In addition, it is interesting that the 17-mer produces a concise group of coronaviruses with 6 coincidences of it. Here, we see that the 17-mer `CAATTATTATAAGAAAG` is shared by the

Hiroshima, Domestic Cat's, Bat's, Wuhan-Hu-1, Usa's sample, and Omicron variants. Here, it is interesting to note that we again have many of the members of the cluster that has the same mark than with 9-mer `TAAACGAAC`.

For instance, always variants related to bats and Wuhan belong to the very same clusters. So that, this study does not iluminate related to the question of if the virus could have been originated by the animals or the lab installed in Wuhan. The question still remains unanswered.

This study also shows that this 9-mer and 17-mer show that strong genomic material is present in new and very virulent variants such as Omicron. Could these marks can be related to virulence and transmision power? Hence, these perspectives deserve to go more in deep with more Bioinformatics tools (2023, Subvariant of Omicron SARS-CoV-2 and its Rapid Global Spread).

5.2 Task E2

Similarly to prior, for each one of the virus types, the given results show the most frequent k-mers for different values of k. In case of HIV-1 (k=7,9); Adenovirus-2(k=9,16); Ebola(k=9,10,15); and Hepatitis-B(k=9,11). These values have been selected trying to produce optimal clustering among the studied genomes. The tables show the sequence, count, and genome where the k-mer was found. The pylogenetic tree by neighbor joining shows the relationships between the different sequences based on their similarities. The running time of the analysis is also reported. Also, the pylogenetic tree shows the relationships between these genome sequences based on their similarity. The running time of the analysis indicates the computational efficiency of the algorithm used to generate these results is directly proportional to the number of sequences involved. Overall, these results provide insights into the genomic diversity of coronavirus and how different sequences are related to each other. These findings could be used for further research on the evolution and transmission of coronavirus.

Also, our biological conclusions are similar. Nonetheless, the strategies for having concise clusters sometimes require to decrease the number of k-mers, as it happened in HIV-1, the results speak clear about understanding the similarities by studying the pylogenetic trees.

First, **HIV-1** shows its 9-mer `AAGAAAAAA` with 3 coincidences that 'HIV-1 strain HIV-1wk from South Korea', 'HIV-1 clone pIIIB from USA', and 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA' followed by `AAAGAAAAA` and `TTAAAAGAA` with 2 coincidences each that 'HIV-1 isolate 01BRRJUD508 from Brazil', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA' and 'HIV-1 isolate 1031 from Philippines', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA' are also close to the main group. The 7-mer `AAGAAAA` with 4 matches is useful to strenghten that 'HIV-1 strain HIV-1wk from South Korea', 'HIV-1 isolate 01BRRJUD508 from Brazil', 'HIV-1 isolate 1031 from Philippines', 'HIV-1 isolate Pt1-M26-NFL-30-PBio from USA' are close to each other too. Observing these patterns, we can see that HIV-1 South Korea, USA, Brazil, and Philipines in that order, share a proximity in this subset of HIV-1 virus type.

Second, **Adenovirus-2** shows its 9-mer `GAGGAGGAG` with 3 coincidences that 'Raccoon adenovirus isolate W-0155-19B', 'Porcupine adenovirus isolate X-24879-18', and 'Skunk adenovirus PB1' live in the same clusters with a string genomic relationship, followed by the 'Human adenovirus 2' being who shares the major proximity to them. The 16-mer | TTTTTGGCGCCAAAAA comes to iluminate also a strng relationship between 'Raccoon adenovirus isolate W-0155-19B', 'Porcupine adenovirus isolate X-24879-18', and 'Skunk adenovirus PB1'. Hence, we can infer that these species of animals are key in potentially the spread of this virus.

Third, **Ebola** is characterized for its conciseness of its trees, and follwing that trend, we have chosen k=9, 15 to get also concise clusters then. Three of its most frequent 9-mers show three clear groups: `GAAGATTAA` shows that 'Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', and 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626' share a strong relationship; followed by `TTAAGAAAA` with the group of 'Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', and 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626'; and by `TAAGAAAAA` with 'Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Zaire ebolavirus isolate Ebola virus H.sapiens-wt/SLE/2014/Makona-S9', and 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626'. It is important to note that these last two sequences difer from each other with just one mutation. That can suggest that we could be in the case of a wrong written nucleotide?. Also, with not 4 but 3 matches `ATTAAGAAA` shows a new close group compounded by 'Ebola virus - Mayinga, Zaire, 1976', 'Zaire ebolavirus isolate Ebola virus/H.sapiens-tc/COD/1976/Yambuku-Ecran', 'Ebola virus isolate Ebola virus/H. sapiens-tc/COD/1995/Kikwit-9510626'. When we take into consideration k=10 with 10-mers `TTAAGAAAAA`, `ATTAAGAAAA` (having only 3 mutations of distance switching similar nucleotides in it) we can also regroup to only two clusters, which is very interesting to understand a macro behaviour of the virus in a broader scope.

Finally, **Hepatitis-B** with 9-mers it shows a very diverse number of leaves. That is why is useful to concentrate in mentioning that these 9-mers produce each one close clusters: `CCCCCACTG` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA'; `CTGTGCCTT` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA'; `TTCACCTCT` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA'; `TCACCTCTG` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA'; `CACCTCTGC` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA'; and close but with less matches (only 2) `GTCTGTGCC` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus HB20-0698 DNA'; and `CTTGGACTC` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus HB20-0698 DNA'.

However, all these diverse situtaion changes if we choose k=11 because now the cluster and

proximity are more brief and concise. Its 11-mer `TTCACCTCTGC` with 'Hepatitis B virus complete genome, genotype A', 'Hepatitis B virus strain #1958.7', 'Hepatitis B virus HB20-0698 DNA' shows what already `CCCCCACTG` also showed but now with more exclusivity. Now, we see that `TTATATAAGAG` shows that 'Hepatitis B virus complete genome, genotype A' is near to those others.

6 Conclusion

This analysis aimed to identify the most frequent k-mers for different values of k for various viruses, including coronavirus, HIV-1, Adenovirus-2, Ebola, and Hepatitis-B, and generate a phylogenetic tree by neighbor joining. The results provide insights into the genomic diversity of these viruses and how different sequences are related to each other, which could be used for further research on the evolution and transmission of these viruses.

7 Distribution of work

This time, the entire work was done by myself. This was an agreement with Gabriel.

8 References

- The Evolution and Biology of SARS-CoV-2 Variants https://perspectivesinmedicine.cshlp.org/content/12/
- The XBB.1.5 ('Kraken') Subvariant of Omicron SARS-CoV-2 and its Rapid Global Spread. Retrieved from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10050278/
- Inhibition of HIV-1 Replication by siRNA Targeting Conserved Regions of gag/pol. Retrieved from: https://www.researchgate.net/publication/6665603_Inhibition_of_HIV-1_Replication_by_siRNA_Targeting_Conserved_Regions_of_gagpol
- Nucleotide 9-mers Characterize the Type II Diabetic Gut Metagenome. Retrieved from: https://www.sciencedirect.com/science/article/pii/S0888754316300155
- Phillip Compeau & Pavel Pevzner. *Bioinformatics Algorithms–An Active Learning Approach. 3rd Edition.* Active Learning Publishers. La Jolla, California. 2018
- Shabir, O. 2021. *How Does the SARS-CoV-2 Genome Compare to Other Viruses?.* News Medical Life Science. https://www.news-medical.net/health/How-Does-the-SARS-Virus-Genome-Compare-to-Other-Viruses.aspx