CMPE 484 - Assignment 2

Spring 2022

Due 20.05.2022, 09:00

Description

In this assignment, we will be working on the genome of the SARS-CoV-2 virus. For this, we will be using the data set curated by Kunal Joshi & Phillip Compeau. For this task, you are expected to load and visualize the data, and construct a phylogenetic tree based on it.

In this assignment, in order to keep our sequences short, we will only be focusing on the gene that encodes the spike protein of SARS-CoV-2 virus (or a subset thereof). Throughout this assignment, when we refer to the *spike protein gene*, we refer to the nucleotide sequence between the positions 21563 and 25384 (inclusive, positions are 0-indexed). The positions are determined by the annotation of SARS-CoV-2 genome.

Some questions below will require a little playing around with the prominent bioinformatics package `Biopython`, its documentation, and/or searching the web. This is a very useful skill to obtain, hence the assignment requiring you to practice it. However, these are not intended to take too much time; if they do, feel free to contact us.

Requirements

- You are expected to use `Python` programming language for this assignment.
- Make sure you have installed the `Biopython` package (version 1.79) for your `Python` interpreter.
- You are expected to submit a single `.ipynb` file (i.e. Jupyter notebook) that is runnable. Name your notebook with your student id (e.g. `2019400XXX.ipynb`). Note that this notebook will be your report as well, so explain your work in the related sections of the notebook.
- Clearly separate answers to different questions and subquestions with headings.
- Remember to use the markdown feature of the Jupyter notebook for your headings and textual answers.
- Late submission policy: You can submit until 25.05.2022, 09:00 with 20% penalty. After this, no submissions will be accepted.

Questions

1. This part of the assignment concerns the loading and the visualization of the data. Download the data from this link and extract it in its current folder structure. The data set includes the sequencing of SARS-CoV-2 taken from 100 randomly selected individuals at six different time points between 2020-11-03 and 2021-12-08 (the data for 2020-11-17 are corrupted). The data set includes both the raw data as well as its aligned version in FASTA format. In this and the following exercise we will use the aligned version of the data.
    a. Refer to the documentation of `Biopython` to read the FASTA file for the aligned version of the sequences from November 27th data set: `HCOV19-ENGLAND-271120-A.fasta`. Format your data into a `numpy` matrix of size 100 x 3822. We will call such a character matrix a nucleotide matrix, with each character representing a nucleotide, an indel, or a nucleotide-related information.

b. In your nucleotide matrix, detect any characters in the sequences other than 'A', 'T', 'C', 'G', and '-', and convert all of them to '-'. (Note: This is not a standard practice, it is done in this case to make your work easier here.)

c. Take the first 25 samples from November 27th data set, and visualize the first 250 nucleotides of the spike protein gene in the form of a color matrix, such that each nucleotide is represented by a different color and empty characters are left white. (Note: See Appendix A below for further explanations on visualizations.)

d. What are your observations regarding the resulting visualization? How similar are different nucleotide sequences? Are there any visible mutations?

e. Using all November 27th spike protein data (of size 100 x 3822) you read and transformed in 1a-b, compute an entropy score for these sequences, by implementing this function yourself.

2. Using the aligned version of the data, extract the spike protein gene from the first 4 patients of the samples from November 3, November 27, and December 8 and format the data into a `numpy` matrix. You should end up with a nucleotide matrix of size 12 x 3822. Repeat the conversion you did in 1b on these sequences.

a. Use percent identity score to compute the pairwise alignment score between all pairs of nucleotide sequences, and use these to fill a pairwise similarity matrix of size 12 x 12. All values in this matrix should be between 0 and 1. Print your pairwise similarity matrix.

b. Convert your pairwise similarity matrix to a pairwise distance matrix by subtracting every element of your pairwise similarity matrix from 1. Print your pairwise distance matrix.

c. Implement the neighbor joining algorithm without using any third-party software* and using it construct a phylogenetic tree of the given 12 samples. Print the tree you constructed in the Newick format with distances and leaf names.

d. Using your Newick formatted tree as an input, visualize the phylogenetic tree for the 12 samples, using the `Phylo` module of `Biopython` package.

e. Using your pairwise distance matrix as input, use the neighbor joining algorithm provided in the `Phylo` module to construct a phylogenetic tree, and visualize it similarly to above. Compare the resulting tree to that of your own implementation, which you visualized in 2d. Is it different from your result? If so, how and why?

f. Extract the spike protein gene from all patients from November 3, November 27, and December 8 and format the data into a `numpy` matrix. You should end up with a nucleotide matrix of size 100 x 3822. Repeat the conversion you did in 1b on these sequences. Use the `Phylo` module to construct and visualize the phylogenetic tree of these sequences.

* For 2c, you are free to use base data structures of `Phylo` such as `BaseTree.Clade` or `BaseTree.distance_matrix` instead of e.g. `numpy.array`. However, these can only be used passively, e.g. to store your data or interim results. Your implementation of the algorithm should still be manual without any core operations outsourced to `Phylo` and be observable in a step by step fashion. Your outputs should still conform to the formats described above also.

Appendix A: Information regarding the visualization in 1c

One way you could do it is through `matplotlib.pyplot.imshow` function. Let's say you have prepared a nucleotide matrix of size *M* x *N*. You cannot proceed to use `imshow` directly with this matrix, as nucleotides are not numerical and `imshow` requires numerical values. But what you can do is to convert that matrix to an array of size *M* x *N* x 3, where you associate each type of nucleotide with some RGB values and empty spaces with white.

When you visualize this 3d array with `imshow`, the result should be a colorful table of size *M* x *N*,

where each nucleotide type is a different color, and indels are white (see example below). The aim of this visualization would be to quickly see how well the sequences from different patients align.

Example nucleotide matrix:

```
array([['A', 'T', ' ', 'G'],
       ['G', 'C', 'T', 'T']], dtype='<U1')
```

Its visualization: