# Assignment 2 - Phylogenetic Inference

**COMP 561 - Computational Biology Methods and Research**

**LE, Nhat Hung**
McGill ID: 260793376
Date: October 10, 2019
Due date: October 18, 2019

Prof. Mathieu Blanchette
Fall 2019

---

## Question 1. (30 points)

To answer this question, you will need to use some online tools implementing some of the algorithms seen in class. These might include:
Blast ( http://www.ncbi.nlm.nih.gov/BLAST )
and
the LIRMM Phylogenetic inference package
http://phylogeny.lirmm.fr/phylo_cgi/index.cgi

If you have never used Blast before, you may find this tutorial useful: http://www.ncbi.nlm.nih.gov/books/NBK1734/

In particular, learn what an E-value is, and what the different Blast versions do (Blastn vs MegaBlast vs Blastp vs TBlastN, etc.)

Context: You are doctor and you have a patient suffering from a mysterious infection. You've extracted DNA from the infected area, sequenced it, and obtained the DNA sequence at http://www.cs.mcgill.ca/~blanchem/561/mystery.fa

a) (15 points) What do you think is the cause of the infection? What is the name of the disease?
Hint: Default parameters for blastn may not result in the identification of very useful hits. Consider changing some of these parameters in order to try to identify hits with good E-values.

### Solution:

**BLAST Algorithms**

| Algorithm | Alignment Type | Description |
|---|---|---|
| blastn | nucleotide-nucleotide | - General purpose - Can align tRNA, rRNA, mRNA and DNA |
| MegaBLAST | nucleotide-nucleotide | - 10x faster than blastn - Align sequences that are **nearly identical** - Useful for large batches of sequences |
| blastp | protein-protein | |
| tblastn | protein-nucleotide | - Translate protein query to nucleotide before aligning with nucleotide databse |

We will use `blastn`, as we're comparing DNA sequences.

Because the query sequence is pathogenic, we suspect the target sequence comes from a **bacteria or virus** ⇒ we choose to search the "**Nucleotide Collection (nr/nt)" database** instead of "Human genomic + transcript".

With word size = 7, and E value = 0.001, we get **100 hits for the Zika virus with up to 83% identity.**

We conclude the patient was infected by a Zika virus.

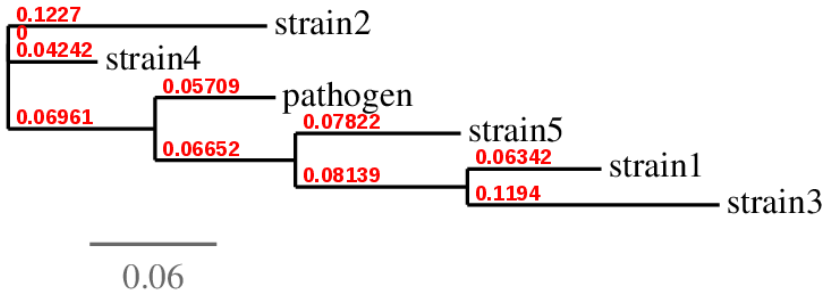
---

b) (15 points) Suppose there exist treatments for various strains of the pathogen. Five known strains exist, with sequences given at: http://www.cs.mcgill.ca/~blanchem/561/strains.fa

Which treatment (i.e. that for which strain) is the most likely to be appropriate for your patient? Use a phylogenetic inference tool such as http://phylogeny.lirmm.fr/phylo_cgi/index.cgi to figure it out. Explain your answer.

### Solution:

The LIRMM tool inferred the following phylogenetic tree for our pathogen and the 5 strains:




The red values represent branch lengths, and accordingly show that **strain 4 is the most identical to our pathogen.**

Therefore, strain 4's treatment is the most likely to be appropriate for our patient.

---

## Question 2. (30 points)

The first algorithm to calculate the parsimony score of a given set of nucleotides at the leaves of a given rooted binary tree was invented by Fitch and Wagner in 1971. The algorithm works as follows. For each node *u* of the tree *T*, define a set *Xu* as follows:

If $u$ is a leaf then
  $X_u = \{x\}$, where $x$ is the nucleotide at leaf u.
  Score(u) = 0

If $u$ is an internal node with children $v$ and $w$, then

  If $(X_v \cap X_w = \emptyset)$ then
    $X_u = X_v \cup X_w$
    Score(u) = Score(v) + Score(w) + 1
  Else
    $X_u = X_v \cap X_w$
    Score(u) = Score(v) + Score(w)

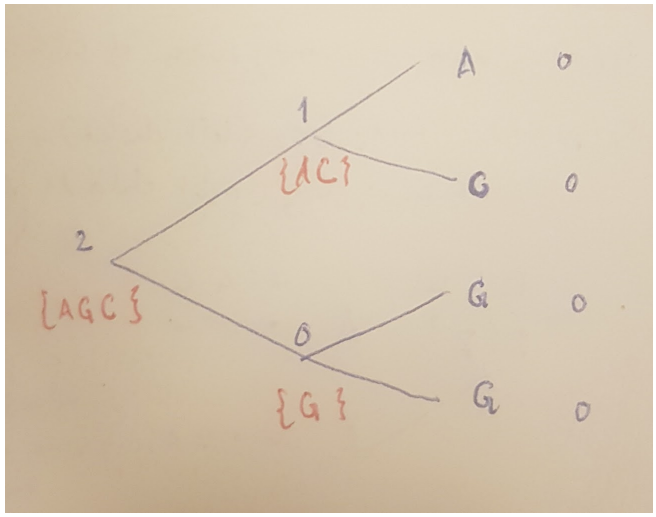
After all the X sets have been computed, starting from the leaves back to the root, Score(root) is the desired parsimony score.

Question: Prove that the algorithm always yields the correct (minimal) parsimony score. Perhaps the easiest (but not the only) way to do this is to show that the Fitch algorithm will always produce the same answer as Sankoff's algorithm, which we can assume is a correct algorithm. Try to be as formal as you can in your proof, but if you don't have experience writing mathematical/algorithmic proofs, just explain your reasoning as
clearly as possible.

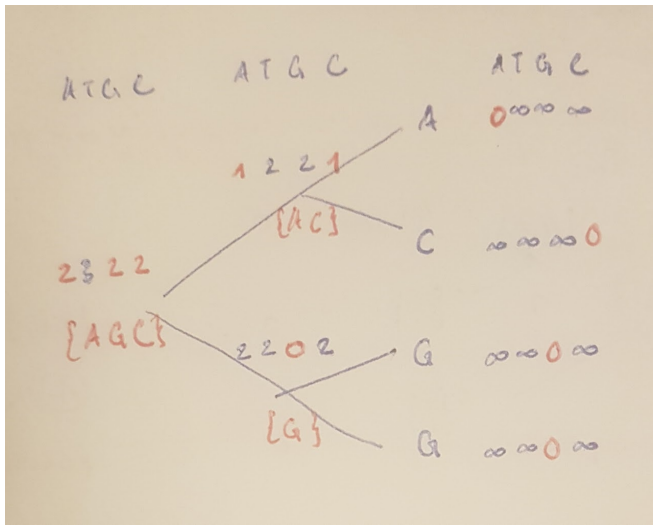
### Solution:

We will show that the Sankoff algorithm with an unweighted parsimony matrix, i.e. match score 0 and mismatch score 1, **gives the same set of optimal nucleotides at each tree vertex as would Fitch's algorithm.**

We can first observe in an example how Sankoff gives the same set of optimal nucleotides as Fitch's:




Fitch




Sankoff

For Sankoff, we have for each node *u*

$$s_x(u) = \text{minimum score of subtree rooted at node } u \text{ if } u \text{ had nucleotide } x$$

We see that nucleotide x is optimal for node *u* if

$$s_x(u) = min_i\{s_i(u)\}$$

Let

$$S_u = \text{set of optimal nucleotides for node } u$$

Then, from above:

$$\begin{gathered} S_u = S_v \cup S_w \quad \text{if } S_v \cap S_w = \emptyset \\ = S_v \cap S_w \quad \text{otherwise} \\ v, w \text{ child nodes of } u \end{gathered}$$

This is identical to Fitch's recurrence.

## Question 3. (40 points)

Phylogenetic trees can be built from non-genetic data, and in fact this was the only type of information available prior to the 1970's, when DNA sequence became possible. Here, you will design and implement an algorithm similar to Sankoff's algorithm, but that will work on quantitative traits (things about a species that can be measured with integers) rather than genetic data. Suppose that you have information about *k* traits for
each species. These traits are measured as non-negative **integers**. For example, for mammals, trait1 might be the length of the forearm (in cm), trait2 might the volume of the skull (in cm^3), trait3 might be the lifespan (in years), etc. As species evolve, their traits change but those changes are generally slow (although there are exceptions). Thus, a parsimony-based phylogenetic inference approach makes sense. The problem we want to solve is the following:

**Input:**

- A set of n species, with, for each species *i*, a vector of *k* integers

$$D_i = (D_{i,1}, D_{i,2}, ..., D_{i,k})$$

  representing the measurements made for the *k* traits

- A phylogenetic tree *T* with leaves labeled with the *n* species.

**Goal:**

Assign a vector *Du* to each internal node *u=n+1, ..., 2n-1* such that

$$\sum_{(u,v)\in E(T)} |D_u - D_v|_1$$

is minimized, where

$$|D_u - D_v|_1 = \sum_{i=1}^{k} |D_u(i) - D_v(i)|$$

a) (25 points) Write the pseudocode of algorithm to solve this problem. You can assume that no trait value will ever exceed a given maximum value *M*.

### Solution:

$$min \sum_{(u,v)} \sum_{i}^{k} |D_u(i) - D_v(i)| = \sum_{i}^{k} min \sum_{(u,v)} |D_u(i) - D_v(i)|$$

The above reduce the problem to *k* small parsimony problems, one for each trait *i*.

Therefore, we will solve them by filling *Du(i)*, for each *i*, for all internal node *u* of the tree *T*. The vectors *Du*'s will be filled according to a Fitch inspired algorithm designed for quantitative values.

The algorithm is as follows:

```
for each trait i:
    for each leaf u:
        Du(i) = { Du(i) } # Initialize leaf values into sets

    choose_optimal_sets(i)
    choose_optimal_values(i)

def choose_optimal_sets(i)
    for each interal node u in in-order traversal (from leaves to root):
        let v, w = u.children

        if Dv(i) and Dw(i) overlap:
```

```
            Du(i) = intersection( Dv(i), Dw(i) )
        else:
            Du(i) = set of integers between Dv(i) and Dw(i)
                  # E.g. {1, 2} and {5, 10} -> {2, 3, 4, 5}

def choose_optimal_values(i)
   for each node u in breadth-first search (from root to leaves):
      if u is root:
          Du(i) = arbitrary value in Du(i)

      else: # u has parent
          let p = u.parent

          if Du(i) contains >1 values:
              Du(i) = value in Du(i) closest to Dp(i)
                    # E.g. Du(i) = {1, 2, 3} and Dp(i) = 5 -> Du(i) = 3
                    #              {1, 2, 3}           = 2          = 2

          else: # Du(i) contains only 1 value
              Du(i) = value in Du(i)
```

---

b) (5 points) What is the running time of your algorithm (using big-O notation)?

## Solution:

In the above algorithm, we have to perform work on *2n - 1* nodes twice, the first time to find sets of optimal values, and the second to choose the optimal *Du(i)* values.

The amount of work for each node depends on the size of its set of optimal values.

The max possible set size being *M*, we obtain a total runtime of **O(*Mn*)**.

**However**, our algorithm actually works without sets: we only used sets for analogy's sake, to be able to easily compare it to Fitch's algorithm.

Instead, we can use **intervals**, each represented by an array containing its 2 boundaries. Then, the work performed at each node will be **constant**.

Therefore, the algorithm can run in **O(*n*) time**.

---

c) (10 points) The problem formulation above is not ideal in cases where the range of values of different traits differs significantly (e.g. the lifespan ranges between 0 and 50 years, whereas the volume of the brain ranges from 0 to 1500 cm^3 ).

**Explain why and propose a modification to the problem formulation that would make it more biologically relevant. You do not need to propose a revised algorithm to go with your revised problem formulation.**

## Solution:

The above formulation overemphasizes nominal differences in values e.g. 50 to 25 years in lifespan would be considered less significant than 1500 to 750 cm^3 brain volume, despite the two changes are identical in scale.

As a result, on the same set of species, a phylogenetic tree focusing on the brain volume trait would look largely different from on focusing on the lifespan trait.

Therefore, for the formulation to be more biologically relevant, we should fill out the *Du*'s vectors with values all scaled down to the same range, e.g. from 0-1, for all quantitative traits.