GS 373 Homework 4

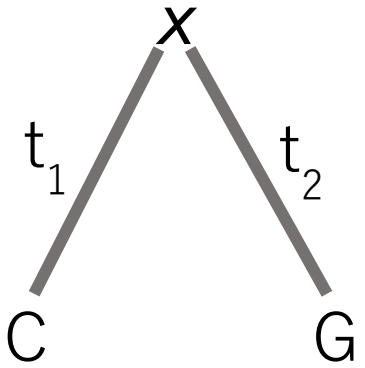
Due May 3rd before 1:30 PM on Canvas

- (100 points): 4 bioinformatics questions (80 points), 1 programming assignment (20 points).
- Submit answers to the bioinformatics questions in a Microsoft Word document or PDF via Canvas. Your answers do not need to contain the text of the questions, but they need to be clearly labeled (e.g., 1a., 3b., etc.)
- Submit the programming assignment as a separate .py file onto Canvas. The script should be able to be directly run by Python.

Bioinformatics Questions (80 points)

1. Tree topologies (20 points)
    a. (8 points) **Draw a rooted tree** which expresses the statement "Chimp (C) and bonobo (B) are most closely related. Humans (H), chimps (C) and bonobo (B) have a common ancestor more recently than their common ancestor with gorillas (G). Orangutans (O) are the outgroup."
    b. (4 points) **Draw an unrooted tree** with the same relationships as above.
    c. (8 points) Starting with your unrooted tree, **draw a different rooted tree** from part a which would also correspond to it. (In other words, show the rooted tree that would be produced if you moved the root to a new location.)

2. Felsenstein's tree-pruning algorithm (20 points)
    Consider this two taxa tree for one site:



C and G are the two observed sequences (of length one nucleotide) and *x* is their (unobserved) common ancestor.

    a. (10 points) **Write out the equation for the likelihood of the data given this tree**. Please expand all summations.
    b. (10 points) Using the likelihood equation from part a and the values below, **calculate the likelihood of the data given this tree**. (Assume that $t_1=t_2=1$)

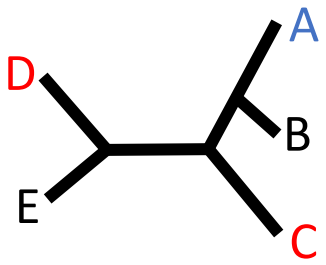
stationary state frequencies	
Prob(A)	0.20
Prob(C)	0.30
Prob(G)	0.30
Prob(T)	0.20

transition probabilities	
Prob(A | A, t=1)	0.80
Prob(A | C, t=1)	0.05
Prob(A | G, t=1)	0.10
Prob(A | T, t=1)	0.05
Prob(C | A, t=1)	0.05
Prob(C | C, t=1)	0.80
Prob(C | G, t=1)	0.05
Prob(C | T, t=1)	0.10
Prob(G | A, t=1)	0.10
Prob(G | C, t=1)	0.05
Prob(G | G, t=1)	0.80
Prob(G | T, t=1)	0.05
Prob(T | A, t=1)	0.05
Prob(T | C, t=1)	0.10
Prob(T | G, t=1)	0.05
Prob(T | T, t=1)	0.80

3. Molecular phylogenetics (20 points)
   a. (10 points) **Name two assumptions** of maximum likelihood phylogenetics and **give an example of how each assumption can be violated**.
   b. (10 points) **Compare and contrast parsimony, distance-based, and maximum-likelihood methods for tree construction**. Write a short paragraph or make a pro/con chart. Feel free to reference lecture slides or Yang and Rannala (2012).

4. (20 points) Tree search space

   How many possible Nearest-Neighbor Interchanges can be performed on the tree below? Draw and label all of the single-interchange neighbor trees. Calculate the parsimony score for each tree based on the color of each leaf (i.e. a single trait with phenotype options of red, blue, and black). Which tree is the most parsimonious?



(20 points) Programming:

For this problem, you will write 3 functions. You can check that each function works as expected by using them to run the code provided below. Submit a final homework4_XX.py script that just contains the definitions of the three functions (it doesn't need to actually do anything when you run it).

1) **A function `euclidean_distance(point1, point2)`** that takes two arguments, both lists of numeric variables of equal length, representing the coordinates of two points. The function should calculate and return the distance between the two points. **It should work for two points in any number of dimensions.** As a reminder, the Euclidean distance between two points in 3-dimensional space (x_1,y_1,z_1) and (x_2,y_2,z_2) is defined as:

$$d = \sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Hint: The function `pow(a,b)` calculates a^b.

2) **A function `manhattan_distance(point1, point2)`** that takes the same arguments as `euclidean_distance()` but instead calculates the Manhattan distance between them. **It should again work for two points in any number of dimensions.** As a reminder, the Manhattan distance between two points in 3-dimensional space is defined as:

$$d = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$$

Hint: The function `abs(a)` calculates the absolute value of a.

3) **A function `calc_distance(point1, point2, method)`** that takes 3 arguments: the first two arguments are the same as in the previous functions and the third is either the string 'euclidean' or the string 'manhattan'. **The function should calculate and return whichever distance the user specifies in the 'method' argument.** You should use your first two functions for this step.

Below is a template for your three functions – you simply need to fill in their calculations and return statements.

```
def euclidean_distance(point1, point2):
     #Calculate and return Euclidean distance here

def manhattan_distance(point1, point2):
     #Calculate and return Manhattan distance here

def calc_distance(point1, point2, method):
     #1) Determine which distance metric the user wants
     #2) Call your previous functions to calculate that distance
```

You can test your functions by running the following code below your function definitions. It should print the same output as below.

```
# points
point1 = [0,2,2,1] #A point in 4-dimensional space
point2 = [1,3,2,2] #Another 4-D point
point3 = [4,3] #A 2-D point
point4 = [2,5] #A 2-D point
```

```
# part 1 and part 2
dist1 = euclidean_distance(point1, point2)
dist2 = manhattan_distance(point3, point4)
print(“Distance 1: {0:.3f}, Distance 2: {1:.3f}”.format(dist1, dist2))

# part 3
euc_dist = calc_distance(point1, point2, "euclidean")
man_dist = calc_distance(point1, point2, "manhattan")
print(“Euclidean distance: {0:.3f}”.format(euc_dist))
print(“Euclidean distance: {0:.3f}”.format(man_dist))
```

Expected output:

Distance 1: 1.732 , Distance 2: 4.000
Euclidean distance: 1.732
Manhattan distance: 3.000