Homework 3: Phylogeny Inference

CSCI 5481, Computational Techniques for Genomics
University of Minnesota
Instructor: Dan Knights

Instructions

- Please turn this assignment in on the course web page.
- There are multiple files to turn in. All text and code should be placed into a single folder with a name like *lastname_exerciseXX*. The folder should then be compressed and submitted as a single archive (.zip or .tgz)
- You must do this work on your own, although you are encouraged to have general discussions with other students. The work you turn in must be your own. Your code will be checked for overlap and for surprising idiosyncrasies in common with other submissions.
- Please write the names of anyone with whom you discussed this assignment at the top of your assignment.
- Please include copious comments in your code. Full credit will only be given for code that is fully commented, meaning that every line that is not completely obvious needs a comment. Partial credit may be given for broken/non-functioning code if the code is well-commented.
- You may use any programming language you wish.
- You may use external packages for reading/writing/storing/traversing trees. For example, the skbio python package TreeNode class would be sufficient, although you are welcome to use other packages. See the example file *trees_in_python_example.py* in the homework files.

Background

This homework assignment is an implementation of the Nei-Saitou neighbor-joining algorithm for phylogeny construction, with estimation of bootstrap support.

Datasets

Download and extract the data: https://canvas.umn.edu/courses/391283/files/folder/Homework03

Note: If you will be using the supplied scripts for visualizing your tree (*hw3-plot-edges.r*), then you will need to install *R* (Google it), then install the "*ape*" package and the "*RColorBrewer*" package by running R, and then entering this command: `install.packages(c('ape','RColorBrewer'))`

The homework folder contains these data files:

hw3.fna
File containing a multiple alignment of 61 bacterial 16S subunit ribosomal RNA sequences.

hw-tip-labels.txt
File containing tab-delimited rows of this format:

```
seqID    Phylum    color
```

There is a subfolder called *example* with a toy DNA file called *example.fna*. This contains examples of correct output for *example.fna*:

example/genetic-distances.txt
Genetic distances (% different) between every pair of sequences in *example.fna*.

example/edges.txt
Correct edges for neighbor-joining tree using R implementation, in preorder traversal order. Your edge order and internal node labels do not need to be exactly the same, but the tip indices should correspond to the order of the input sequences in the fasta file (e.g. tip 1 is the first sequence in the input file, tip 2 is the second sequence, and so on).

example/bootstrap.txt
Example bootstrap support values for the 5 internal nodes, labeled with the same node index used in *edges.txt*.

tree.pdf, *tree-bootstrap.pdf*
Example PDF plot of tree showing colored tips and bootstrap support for internal nodes.

Input and Output Format:

Your program **for questions 1-2** should take one command line argument specifying the name of a sequence file. The command line should be of the form ~~*programName -i sequence.fna*~~ *programName sequence.fna* (it is acceptable to invoke java, python, R, or another program as part of programName). Your program should output two files:

genetic-distances.txt
A tab-delimited table of pairwise distances between all sequences, following the format in the *example* folder.

edges.txt
This file is tab delimited. Each row describes an edge in the tree. The first column is the ancestor node; the second column is the descendant node; the third column is the edge length. Edges should be in preorder traversal order choosing any internal node as the root. Tips must be indexed starting at 1, with 1 corresponding to the first sequence in the FASTA file, 2 corresponding to the second sequence, and so on. The internal nodes should begin numbering at the root with *ntips + 1* and the numbers should increase according to preorder traversal.

For question 7, your program should write the bootstrap confidence values for the internal nodes in your original tree (from question 2) to a text file, where the first column is the internal node index from your original tree, the second column is the bootstrap fraction/confidence for that internal node, and the two columns are separated by a "tab" character. See bootstrap.txt in the example folder for an example.

Problems

1. (20 points): Read in the given FASTA file *hw3.fna*. Calculate the genetic distance (% dissimilarity) between every pair of sequences, and write this to a tab-delimited file with rows and columns labeled by the sequence identifiers. For pairwise dissimilarity calculations, you may count a gap in both sequences as a similarity. Write the output to *genetic-distances.txt* as described above.

2. (30 points): Implement Nei-Saitou neighbor joining as described on Wikipedia (https://en.wikipedia.org/wiki/Neighbor_joining) and/or in class notes. Provide extensive comments in your code. We suggest that you store your tree in this data structure, but you can do it however you want:

> `edges,` A 3-column matrix with column 1 representing an ancestor node, column 2 representing the descendant node, and the final column representing the edge length. Tips should be indexed starting at 1. Choose an arbitrary internal node to be the root. The internal nodes should begin numbering with the root as *ntips + 1*.

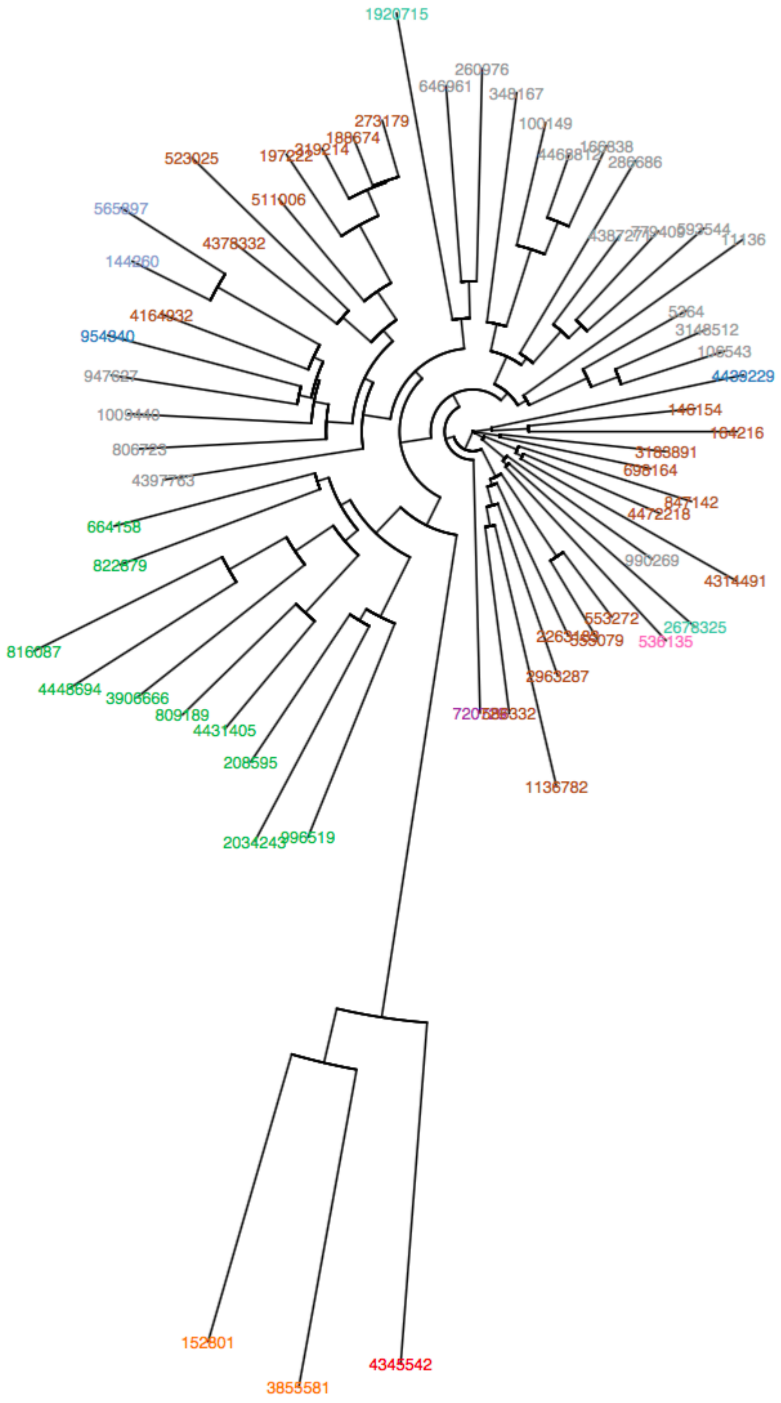
Write the tree to an output file *edges.txt* as described above.

3. (15 points): Find a way to visualize your trees from step (3). If you made the output correctly for step (3), you can use the included R script (after installing the “ape” and the “RColorBrewer” package as described above):

```
Rscript hw3-plot-edges.r edges.txt hw3-tip-labels.txt tree.pdf
```

Colors for the tips are provided in *hw3-tip-labels.txt*. Colors are nice but optional; you must include labels for your tips. ASCII art is also acceptable if you want to generate your own tree visualization.

Here is what the correct Nei-Saitou tree looks like when visualized with the provided R script:


1920715
260976
646961
348167
273179
188674
219214
197222
523025
100149
186838
4468812
286686
511006
565897
4378332
4387271
779409
592544
11136
144260
4164932
5264
3148512
954840
106543
4438229
947627
146154
1009440
164216
806723
3185891
698164
4397763
847142
4472218
664158
822679
990269
4314491
816087
553272
2678325
2263188
555079
536135
4448694
3906666
2963287
809189
4431405
720788
536332
208595
1136782
2034243
996519
152801
3855581
4345542

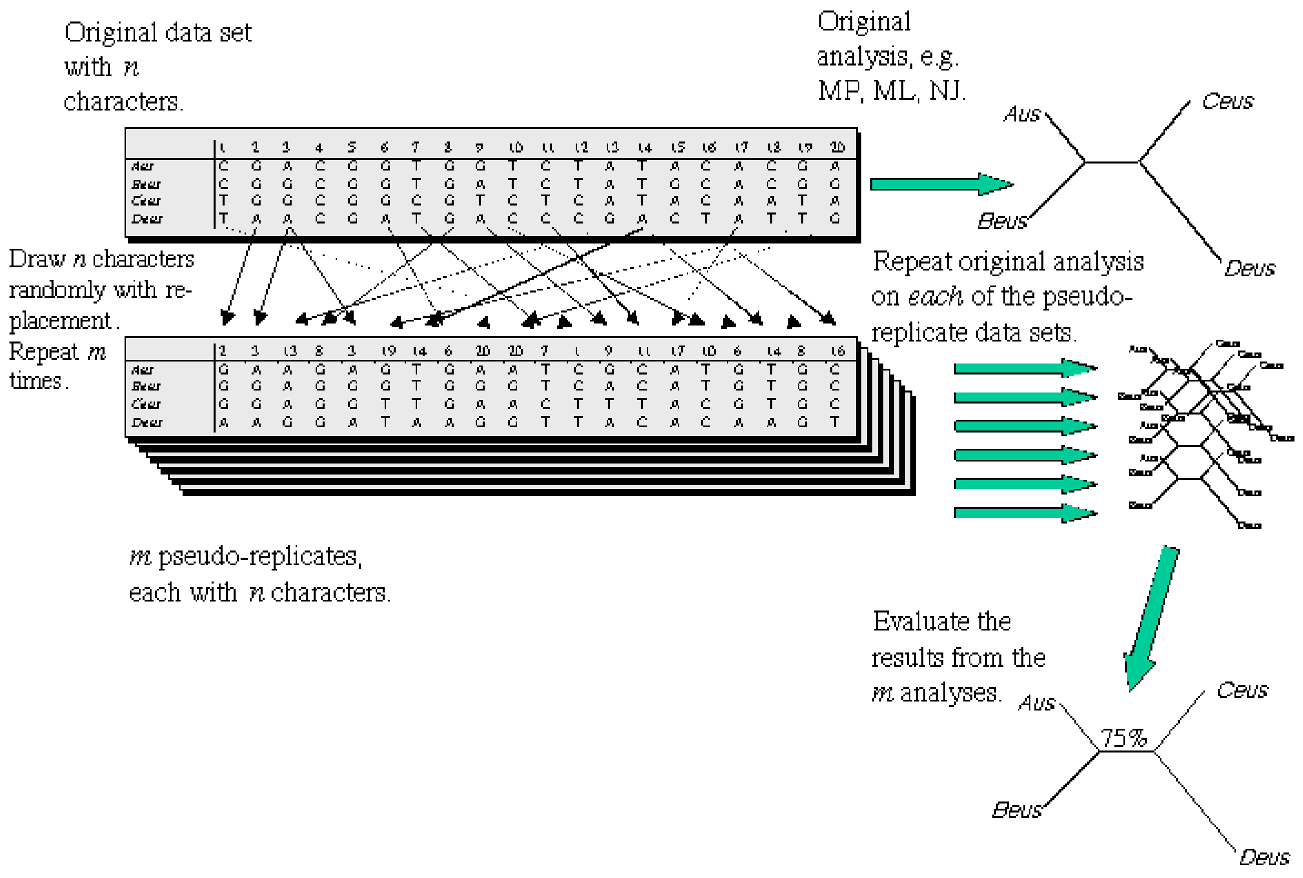
4. (2 points): Why are tips of similar color mostly clustered together?

5. (3 points): Name two distinct reasons why the clustering by color is not perfect.

6. (15 points): Generate bootstrapped trees. A common way to assess confidence in the structure of an inferred phylogeny is to subsample the input DNA, rebuild the tree from the subsampled data, compare the new tree to the original tree, and repeat many times. If the DNA is subsampled with replacement, this process is called bootstrapping.

In this problem, you will write code to generate bootstrapped samples of the input DNA, and build a bootstrapped tree from each of them. Your code should take as a parameter the number of bootstrapped samples to generate, for example, 100.

Each bootstrap sample of the DNA sequences should contain a random rearrangement of the original DNA columns selected by repeated sampling with replacement until there are as many columns (characters) as in the original input. This means that some columns will appear more than once in a bootstrap sample. Here is a nice depiction of the resampling that someone shared with me (unfortunately I lost the attribution for the figure):


Original data set with *n* characters.
Original analysis, e.g. MP, ML, NJ.
Aus
Beus
Ceus
Deus
Draw *n* characters randomly with replacement. Repeat *m* times.
Repeat original analysis on *each* of the pseudo-replicate data sets.
m pseudo-replicates, each with *n* characters.
Evaluate the results from the *m* analyses.
75%

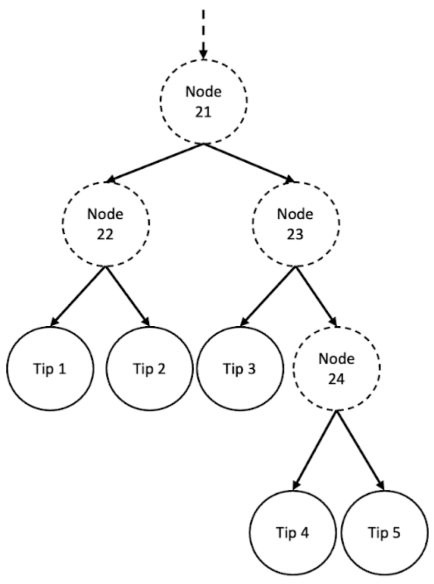
The output of the program should be a folder containing the *n* bootstrapped DNA samples and the *n* bootstrapped trees created from them. However, you do not need to turn in the actual DNA samples and trees (see Deliverables).

7. (10 points) Calculate bootstrapped confidence for each internal node in the original tree, using 100 bootstrapped trees. For each internal node, get a list of the tips that are below it on the left, and below it on the right (left and right children chosen arbitrarily). This is the partitioning of tips by that internal node.

Count the fraction of bootstrap trees in which the exact same partitioning was made by some internal node. That is the bootstrap confidence for the given internal node in the original tree. You must print the bootstrap values to a text file in a tab-delimited table, where the first column is the internal node index from your original tree (as shown in your edges file from step (2) above), and the second column is the bootstrap fraction. See *bootstrap.txt* in the *example* folder for an example.

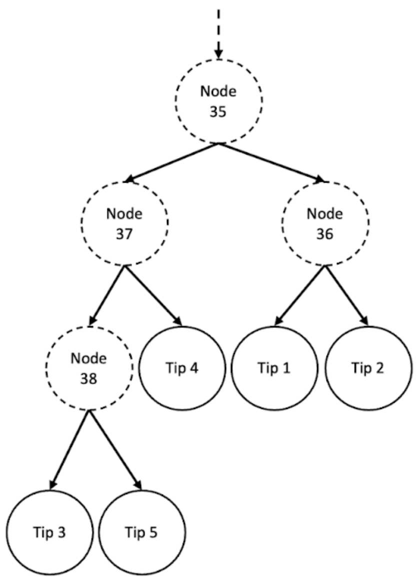
Consider this example subtree from the original tree, showing the partitioning of tips below internal node 21, and example subtrees from two bootstrapped trees. Note that the numbering of the internal nodes does not need to correspond from tree to tree; only the tip labels matter. In this example, internal node 35 in bootstrapped tree 1 has the same partitioning as as the original tree, even though the left/right order and the subtree topology is different; internal node 19 in bootstrapped tree 2 does not have the same partitioning, because Tip9 has been added to the subtree. If these were the only bootstrapped trees sampled, then Node21 would be given 50% bootstrap support, because its exact partitioning was found in half of the bootstrapped trees.

Part of the original tree:



Partitioning at Node21:
[Tip1, Tip2], [Tip3, Tip4, Tip5]

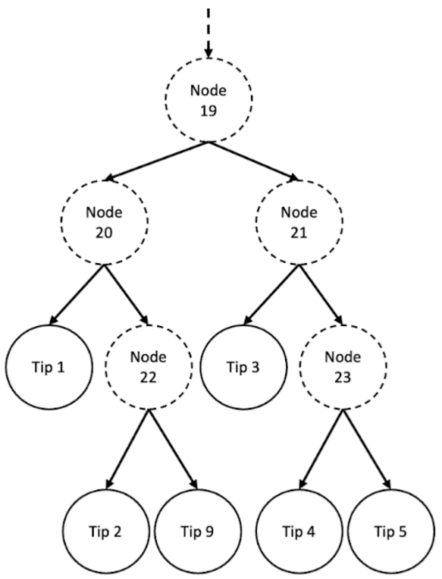
Part of bootstrapped tree 1:



Partitioning at Node35:
[Tip3, Tip4, Tip5], [Tip1, Tip2]

Same partitioning

Part of bootstrapped tree 2:



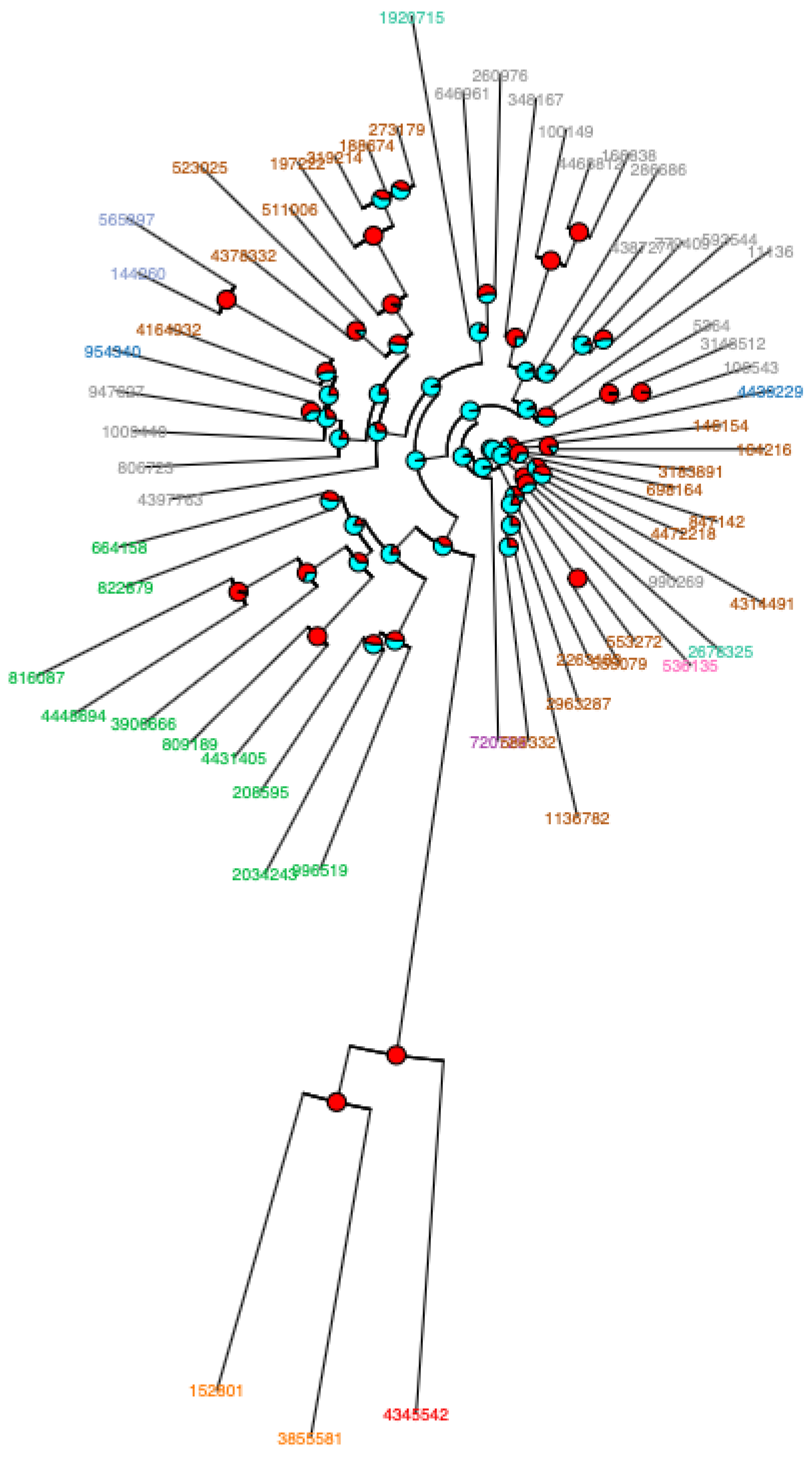
Partitioning at Node19
[Tip1, Tip2, Tip9], [Tip3, Tip4, Tip5]

Different partitioning

You can also optionally plot the bootstrap confidence using the supplied R script `hw3-plot-edges.r` (red is fraction with bootstrap support):

```
Rscript hw3-plot-edges.r edges.txt hw3-tip-labels.txt bootstrap.txt
tree-bootstrap.pdf
```

This an example of a correct tree (100 bootstrap trees) showing bootstrap support for internal nodes. Your bootstrap values will differ slightly:



8. (5 points): Based on the bootstrap tree pictured above, why is bootstrap support generally higher for the internal nodes closest to the tips, and lower for the internal nodes closest to the root?

~~**Deliverables**~~

- ~~Source files (your code for Steps 1, 2, 6, 7).~~
- ~~Distances file *genetic-distances.txt*, from step 1~~
- ~~Edges file *edges.txt* from step 2~~
- ~~Readme file *readme.txt* explaining how to use your code (text)~~
- ~~Visualization of tree from step 3 *tree.pdf*~~
- ~~Bootstrap values *bootstrap.txt* for Step 7 and optional visualization *tree-bootstrap.pdf*~~
- ~~Text file *written_responses.txt* containing responses to questions 4, 5, and 8.~~

~~All files and source code should be added to a folder with your x500 username as the name of the folder. Then, zip this folder and upload it on Canvas.~~

Deliverables

1. Please submit to the "**HW3 Programming**" assignment on Gradescope with the following items
   a. A python file titled **`neighbor_joining.py`**, and any other python files from which **`neighbor_joining.py`** imports packages, etc.
      i. Please note in your code the names of the people with whom you discussed the assignment.
      ii. If you are not using python, please submit your non-python code instead, and a readme file (text). The readme file should contain instructions on how to compile and run the program. We will download and manually run + grade your code.
   b. You don't need to submit your output files, the autograder will run your code and generate the neighbor joining outputs, and will verify that they are correct.
   c. The autograder will run this command to grade the hw:
      1. `python3 neighbor_joining.py data/hw3.fna`
2. Please submit to the "**HW3 Written**" assignment on Gradescope the following items
   a. A single PDF with your responses to the other questions (3, 4, 5, 8)

To summarize, your submission should include the following files: **neighbor_joining.py, any other python files, HW3_Written.pdf,** README.txt (optional)

No need to submit to Canvas if you have already submitted all the necessary files to Gradescope.