# 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 all students with whom you discussed the assignment at the top of your code.
- 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/333064/files/folder/Homework03

Note: If you will be using the supplied scripts for visualizing your tree (*hw3-plot-edges.r*, *hw3-plot-newick.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/tree.tre*
Correct solution output tree in NEWICK format. Your node order does not need to be exactly the same.

*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-newick.pdf*, *tree-bootstrap.pdf*
Example PDF plot of tree showing colored tips and bootstrap support (bonus) for internal nodes. Note that *tree-newick.pdf* and *tree.pdf* are identical. This is because the same tree is represented in the Newick format *tree.tre* and the edges format *edges.txt*

**Input and Output Format:**

Your program for questions 1-3 should take one command line argument specifying the name of a sequence file. The command line should be of the form *programName sequence.fna* or *programName -i sequence.fna* (it is acceptable to invoke java, python, R, or another program as part of programName, but only take one argument as input).

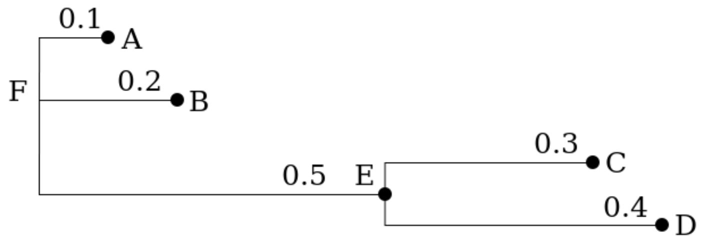
Your program should output three files:

*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.

*tree.txt*
This file is in NEWICK format with all edge distances and with only tips named. For example, The following tree would be encoded (A:0.1,B:0.2,(C:0.3,D:0.4):0.5); :



**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*.

2. (20 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. I 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*.

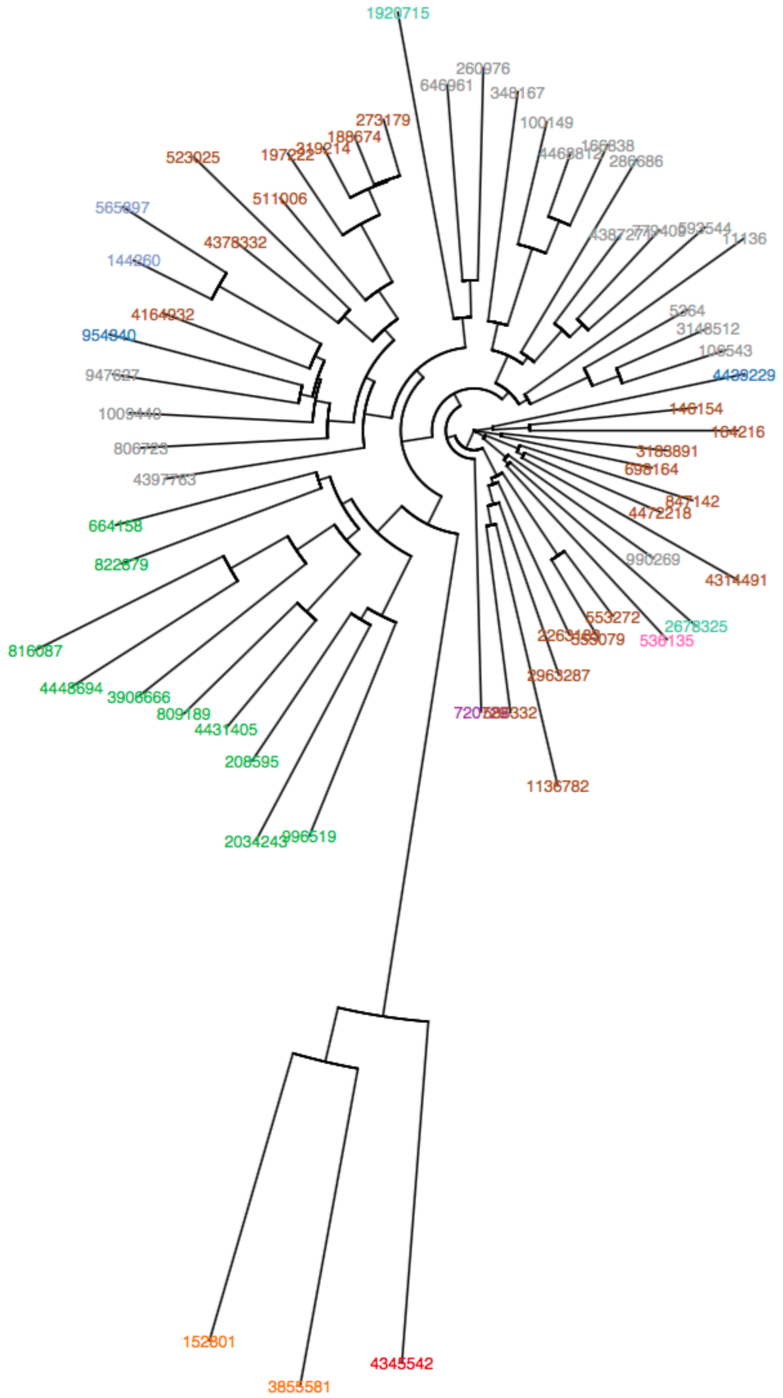
3. (20 points): Generate the two output files described above for your tree (edges matrix *edges.txt* and NEWICK tree file *tree.tre*). For the edges matrix you will need to perform a preorder traversal. For the NEWICK file you will need to perform a postorder traversal. You are not required to use recursion, although it is a common approach. You can also use an external package like the skbio python package TreeNode class to write your tree in NEWICK format.

4. (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
Rscript hw3-plot-newick.r tree.tre hw3-tip-labels.txt tree-newick.pdf
```

There are also NEWICK-based viewers online. 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.

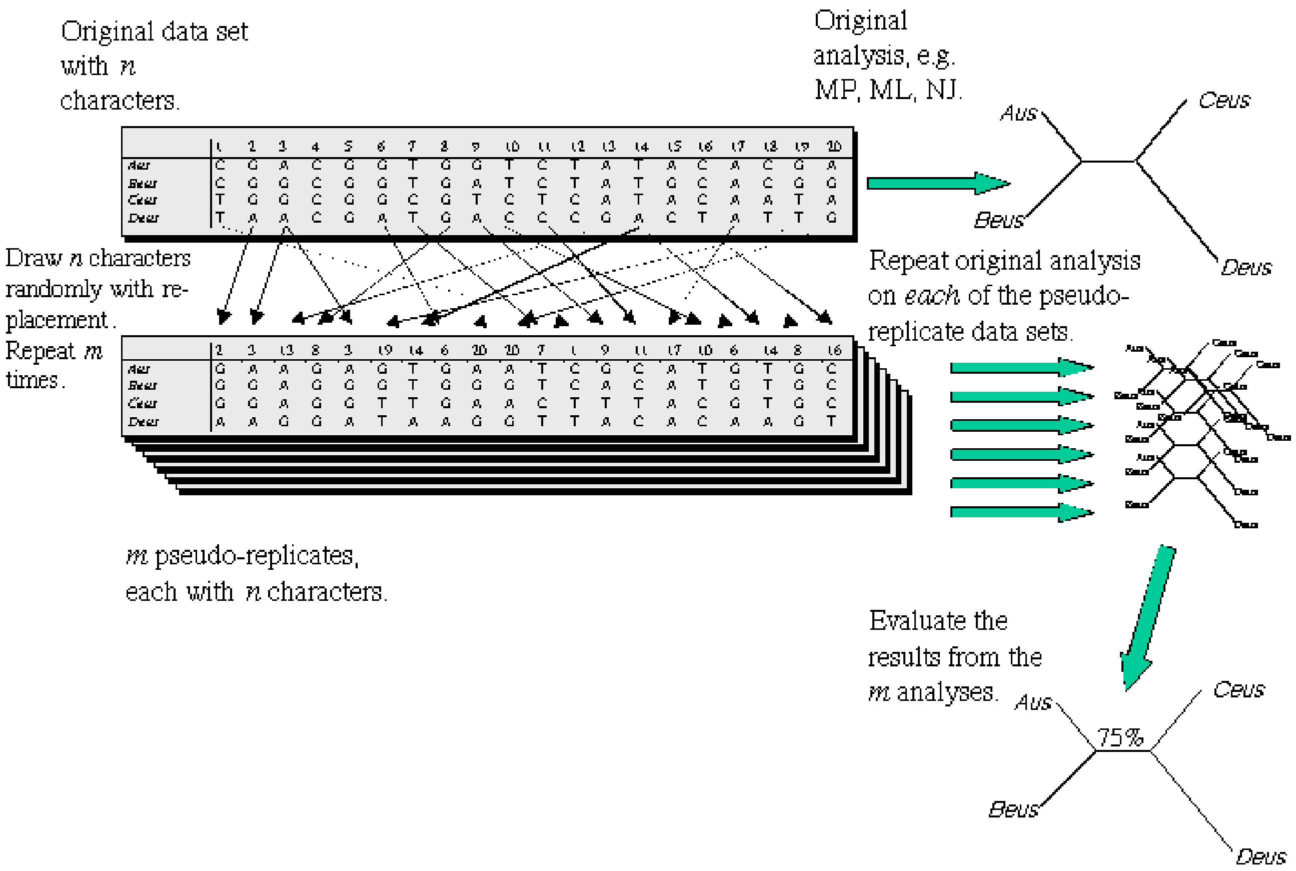
Here is what the correct Nei-Saitou tree looks like:



5. (2 points): Why are tips of similar color mostly clustered together?

6. (3 points): Name two distinct reasons why the clustering by color is not perfect.

7. (15 points): Perform 100 inferences of the tree in step (3) using bootstrap samples of the input DNA. 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 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):



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Aus | C | G | A | C | G | G | T | G | G | T | C | T | A | T | A | C | A | C | G | A |
| Beus | C | G | G | C | G | G | T | G | A | T | C | T | A | T | G | C | A | C | G | G |
| Ceus | T | G | G | C | G | G | C | G | T | C | T | C | A | T | A | C | A | A | T | A |
| Deus | T | A | A | C | G | A | T | G | A | C | C | C | G | A | C | T | A | T | T | G |

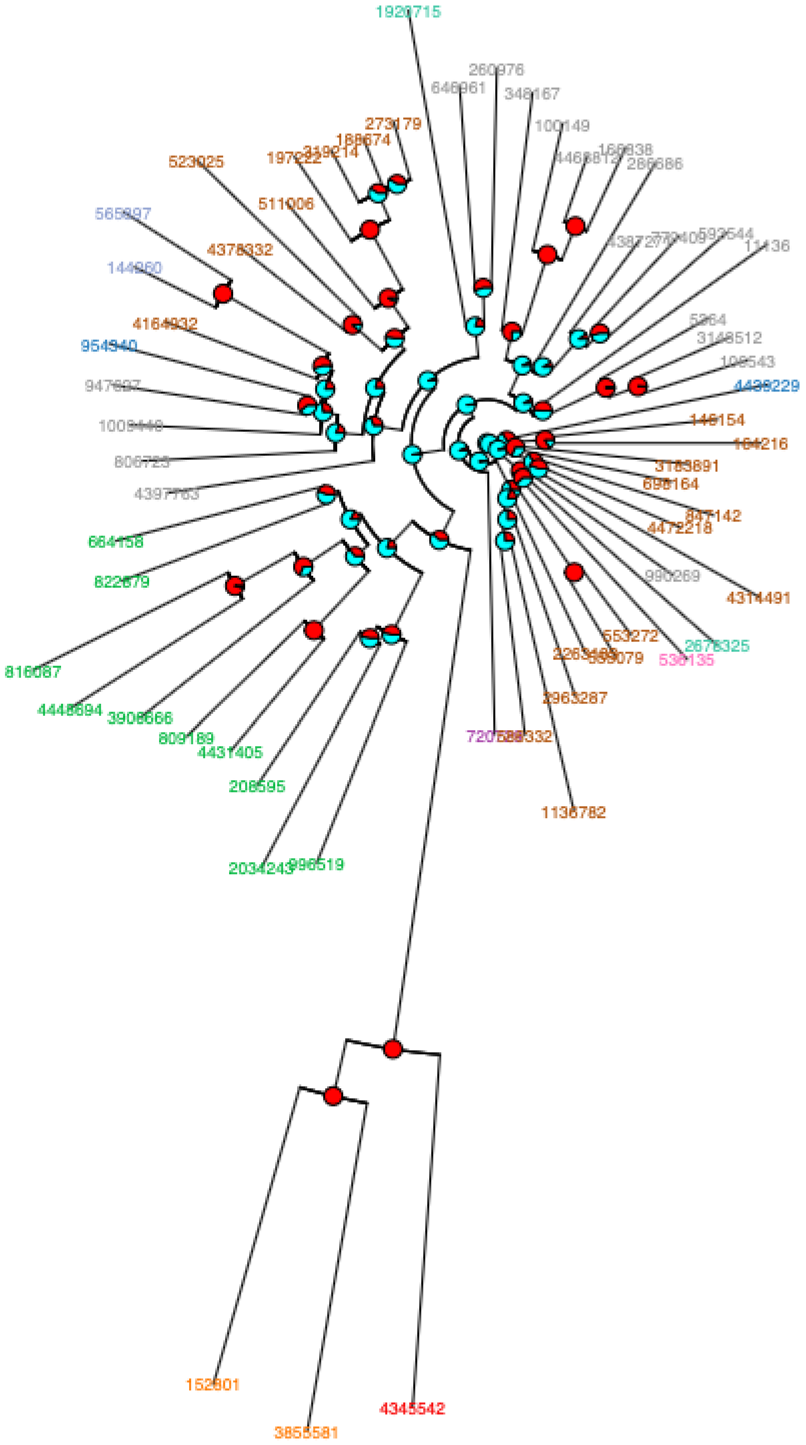
| | 2 | 3 | 13 | 8 | 3 | 19 | 14 | 6 | 20 | 20 | 7 | 1 | 9 | 11 | 17 | 10 | 6 | 14 | 8 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Aus | G | A | A | G | A | G | T | G | A | A | T | C | G | C | A | T | G | T | G | C |
| Beus | G | G | A | G | G | G | T | G | G | G | T | C | A | C | A | T | G | T | G | C |
| Ceus | G | G | A | G | G | T | T | G | A | A | C | T | T | T | A | C | G | T | G | C |
| Deus | A | A | G | G | A | T | A | A | G | G | T | T | A | C | A | C | A | A | G | T |

For each node in the original tree, get a list of the tips that are partitioned below it. Count the fraction of bootstrap trees in which the exact same partition was made by some internal node. That is the bootstrap confidence for the given internal node. 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 (3) above), and the second column is the bootstrap fraction. See *bootstrap.txt* in the *example* folder for an example.

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 is the correct bootstrap tree. 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, 3, 4, 7).
- Distances file *genetic-distances.txt*, edges file *edges.txt*, tree file *tree.tre* from steps 1, 2, 3
- Readme file *readme.txt* explaining how to use your code (text)
- Visualization of Step 4 tree *tree.pdf*
- Bootstrap values *bootstrap.txt* for Step 7 and optional visualization
- Text file *written_responses.txt* containing responses to questions 5, 6, 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.