**Computer Lab 6 – Tree Building (Part 2)**

**Introduction to Models of DNA Sequence Evolution and Maximum Likelihood**

**Conservation Genetics (BIOL 4174 / 5174)**

Part I – Models of DNA Sequence Evolution

Before running a model-based method of tree-building, we must first determine which model is most appropriate for our data set. For this purpose we will use jModelTest 2.1.4. jModelTest is available free of charge from <http://code.google.com/p/jmodeltest2/> for Mac, Windows, and Linux. It also requires the latest version of Java to run (<http://www.java.com/en/>).

In the interest of time, we will run only a subset of the models that can be tested by jModelTest. This will not have any substantial impact on our analysis.

Download the lab 6 assignment file from Blackboard and unzip it in your personal directory on the desktop. First, load your **helo-atp6.phy** file into jModelTest (File 🡪 Load DNA Alignment).

**Testing a series of models**

1. Go to Analysis 🡪 Compute Likelihood Scores.
2. Change the “Number of substitution schemes” to 3.
3. Also select “Fixed BIONJ-JC” under “Base tree for likelihood calculations.” This step is necessary to fix the tree used for the model testing so that a hierarchical likelihood ratio test can be performed.
4. Now click “Compute Likelihoods.” After a few minutes the program will finish running.

**Calculating AIC, BIC, and hLRT**

1. Go to Analysis 🡪 Do AIC Calculations….
2. Click “Do AIC Calculations.”
   * Repeat steps 5 and 6, except select “BIC calculations” and “hLRT calculations.” Leave all settings at default for each of these tests.
3. Save your results by going to “Edit 🡪 Save Console.” Accept the default file name and click “save.”
4. Clear the console by choosing “Edit 🡪 Clear.”
5. Repeat steps 1 through 8 using the **helo-atp8.phy** and **helo-atp86.phy** files.

**Answer the questions in the homework document that pertain to jModelTest**

Part II – PhyML

PhyML is a program for computing maximum likelihood trees, and is freely available at <http://www.atgc-montpellier.fr/phyml/>. The program is efficient, but some calculations are still time consuming. PhyML has been pre-installed on all of the lab computers to utilize their multicore processors to parallelize the bootstrap procedure. This program relies on the Phylip file format (.phy).

**Calculating Maximum Likelihood Trees**

First we will build a tree for ATP8 using the results from jModeltest. Open your saved console file for ATP8 and scroll through it until you find the selected model information for the hLRT. It should look something like this:

Model selected:

Model = GTR+G

partition = 012345

-lnL = 1715.0289

K = 80

freqA = 0.3930

freqC = 0.2999

freqG = 0.0788

freqT = 0.2284

R(a) [AC] = 0.5475

R(b) [AG] = 2.5820

R(c) [AT] = 0.5521

R(d) [CG] = 0.2436

R(e) [CT] = 2.8306

R(f) [GT] = 1.0000

gamma shape = 0.6690

For today’s lab, the important parts are the **model** (GTR+G), **alpha value** for the rate parameter (gamma shape = 0.6690) and p-inv, the **proportion of invariable sites** (not present in this file, but you may see this parameter selected by jmodeltest for other parts of today’s lab). Other parameters can be set in PhyML, but will be ignored for today’s lab.

Open a terminal window and switch to the directory where your lab 6 files are located. Type the following command to execute the program on the helo-atp8.phy file:

mpirun –n 4 phyml –i helo-atp8.phy –m GTR –a 0.669 –b 100

**An explanation of this command:**

* The parallelized bootstrapping features of PhyML rely upon a common external program named **mpirun**.
  + **-n [number]** tells the program to use a certain number of processor cores for bootstrapping (four cores, in this example).
* The command **phyml** tells mpirun to run PhyML.
  + **-i [file]** specifies the input file (helo-atp8.phy).
  + **-m [model]** specifies the model (GTR, or general time reversible model).
  + **-a [number]** tells PhyML that the gamma rate parameter is being used in the model, and sets the alpha value for this parameter (0.669 for this example).
  + **-b [number]** tells the program to perform a certain number of bootstraps (100 for this example).

**PhyML Output**

You can now sit back and wait for the program to finish running. While waiting, read ahead to the “**generating more PhyML trees**” section and try to figure out the commands for the next two trees you will build. When the program finishes running, it will have created four files in the same directory where your helo-atp8.phy file is stored. These are:

* **helo-atp8.phy\_phyml\_stats.txt**: This file contains the model, parameters, and other settings used by PhyML to estimate the tree for the helo-atp8.phy dataset.
* **helo-atp8.phy\_phyml\_boot\_stats.txt**: This is similar to the previous file, except the parameters that were used for each bootstrap replicate are shown.
* **helo-atp8.phy\_phyml\_boot\_trees.txt**: This file is a collection of all of the bootstrap replicate trees that were generated.
* **helo-atp8.phy\_phyml\_tree.txt**: This file is the best maximum likelihood tree estimated by PhyML containing bootstrap values for each node.

**Viewing Phylogenetic Trees**

When PhyMLhas finished running, the **helo-atp8.phy\_phyml\_tree.txt** file can be viewed graphically in FigTree. Open FigTree, go to File 🡪 Open… and navigate to your lab\_6 directory. Open the “helo-atp8.phy\_phyml\_tree.txt” file.

1. When prompted to enter the labels on the nodes/branches, enter “bootstrap” in this space.
2. Root the tree by S\_punctatus. Click on this taxon, then click the “Reroot” button.
3. To make the tree easier to read, click the “Tree” menu at the top of your screen and go to “Increasing Node Order.”
4. Display bootstrap values on your tree by checking “Node Labels” in the left pane. Then expand this option, and select “bootstrap” from the “Display” drop-down box.
5. Expand the “Trees” menu in the left pane. Check the “Transform branches” box and select “proportional” in the associated drop-down menu to make bootstrap values more easily readable.
6. Finally, export your tree as a PDF (File 🡪 Export PDF…) with the filename **phyml\_atp8.pdf**. Save it in your “lab\_6/exported\_trees” folder.

**Generating more PhyML trees**

Now generate trees for your helo-atp6.phy and helo-atp86.phy files using PhyML. Build these trees using the settings you obtained from the hLRT in jModelTest for each file.

* You can do this by modifying the previous command for running PhyML.
  + You will have to **change** the following for each command:
    - Input file name ( –i )
    - Alpha value ( –a )
  + You will have to **add**:
    - The flag for the proportion of invariable sites parameter (flag -v), and its associated value
      * Get the p-inv value from your saved jModelTest console.
* Export trees for each result from FigTree as described in the “Viewing Phylogenetic Trees” instructions on page 3 of this lab. Name your outputs **phyml\_atp6.pdf** and **phyml\_atp86.pdf**. Make sure these are in your “lab\_6/exported\_trees” folder.

**What command did you issue to run PhyML** **on the helo-atp6.phy file? Enter this command in your homework document (Question 1 under Maximum Likelihood Treebuilding).**

**What command did you issue to run PhyML** **on the helo-atp86.phy file? Enter this command in your homework document (Question 2 under Maximum Likelihood Treebuilding).**

Part III – RAxML

RAxML v8.2 is a program for computing maximum likelihood trees, and it is freely available at <http://www.exelixis-lab.org/>. Versions are available for Windows, Mac, and Linux operating systems. The program is very powerful and complex, offering multiple methods for bootstrapping (both “normal” and “fast” bootstrap). However, it only implements variations of the GTR nucleotide substitution model. This program has also been installed on the lab computers to take advantage of multicore processors for parallelization of bootstrap analysis. Like PhyML, this program also relies on the Phylip file format for input.

Although all programs that build trees using Maximum Likelihood methods rely on the same basic principles, some use different algorithms to implement those methods. RAxML will be used to demonstrate some of the differences in results that can arise from analyses using different programs, and different algorithms within programs. RAxML implements two different bootstrapping algorithms for determining support – a “normal” bootstrap analysis, and a “fast” bootstrap analysis which is a necessity for very large datasets.

**Normal Bootstrap Analysis**

First you will generate a tree using the “normal” bootstrap analysis for the helo-atp86.phy file in RAxML. This requires three separate commands. Like PhyML, this program also uses flags at the command line to control different options. An explanation of these flags will be provided at the end of each section.

Issue the following command to conduct the likelihood search for the best tree:

mpirun -n 4 raxmlMPI -p 1234 -s helo-atp86.phy -n mlsearch -m GTRGAMMAI -# 10

When the previous command finishes running, issue the following command to conduct the “normal” bootstrap analysis:

mpirun -n 4 raxmlMPI -p 1234 -s helo-atp86.phy -n boot -m GTRGAMMAI -b 1234 -# 100

When bootstrapping completes, issue the following command to project the bootstrap values on the best tree.

raxml -f b -t RAxML\_bestTree.mlsearch -z RAxML\_bootstrap.boot -m GTRGAMMAI -s helo-atp86.phy -n boottree -T 4

This creates a tree file named “RAxML\_bipartitionsBranchLabels.boottree” which contains your best tree with the bootstrap values projected on it. This file cannot be read by FigTree in its current format, so I have written a perl script which converts it to a format readable by FigTree. You may have to make the script executable (chmod u+x) to run it.

Run this script by typing the following:

./raxml2figtree.pl RAxML\_bipartitionsBranchLabels.boottree

This will generate a new file named “RAxML\_bipartitionsBranchLabels.boottree.figtree.” Open this tree in FigTree and export it as described earlier in the “Viewing Phylogenetic Trees” section with the file name “raxml\_boottree\_atp86.pdf.” Make sure it is saved in your “exported\_trees” folder.

**Fast Bootstrap Analysis**

A single command is required to conduct the fast bootstrap analysis:

mpirun -n 4 raxmlMPI -p 1234 -f a -s helo-atp86.phy -n fastboot -m GTRGAMMAI -x 1234 -# 100

This creates a tree file named “RAxML\_bipartitionsBranchLabels.fastboot” which contains your best tree with the bootstrap values projected on it. Run the perl script to convert it to a format readable by FigTree by typing the following:

./raxml2figtree.pl RAxML\_bipartitionsBranchLabels.fastboot

This will generate a new file named “RAxML\_bipartitionsBranchLabels.fastboot.figtree.” Open this tree in FigTree and export it as described above with the file name “raxml\_fastboot\_atp86.pdf.”

**An explanation of the flags used by RAxML for analysis:**

* **-# [number]** is used to specify the number of different starting trees to test when searching for the best maximum likelihood tree, or the number of bootstrap inferences to conduct.
* **-b [number]** is used to specify the random number seed for the bootstrap analysis. This is used for repeatability of results.
* **-f [algorithm]** is a command to control the behavior of the program. Using -f b will draw the bootstrap values onto a tree specified by the –t flag. Using -f a will tell RAxML to conduct the rapid bootstrap analysis and search for the best maximum likelihood tree in a single run. There are several other algorithms for this flag as well – see the RAxML instruction manual for details.
* **-m [model]** is used to specify a model. For nucleotide data, the program only uses variations of the GTR model.
* **-n [name]** is used to specify the name of the run. This will be used for naming output files. A new name has to be specified for each run of the program (it refuses to overwrite previous runs).
* **-p [number]** is used to specify the random number seed. This is used for repeatability of results.
* **-s [file]** is used to specify a DNA sequence alignment as input
* **-t [file]** is used to specify an input file containing a single tree.
* **-T [number]** is used to specify the number of processor cores to be used in parallelization of the bootstrap procedure.
* **-x [number]** turns on the rapid bootstrapping algorithm, and specifies the random number seed for that algorithm.
* **-z [file]** is used to specify an input file containing multiple trees.

RAxML has many more options as well. If you are interested in using this program on your own data, these options can be found in the most recent version of the RAxML manual which is available at the program’s website.

Part III – Questions

Answer the remaining questions in the “lab\_6\_homework.docx” file. Submit this file via Blackboard after answering all questions.