Computational Exercise in PAUP*

We will be using the software program called PAUP* (Phylogenetic Analysis Under Parsimony *and other methods) written by David Swofford. PAUP* is an old program that has been around for decades now (v3 in 1989). As its name suggests the original version only performed parsimony. For many years, PAUP* v4.0b10 was commercial software distributed by Sinauer Associates. However, recently the license has expired and PAUP* will now be released as an open-source software with "test" versions available now. While a variety of alternative likelihood and parsimony software programs exist, few have as reliable a likelihood estimation as PAUP*. There are GUIs available for some versions, but we will be using a command line version.

Logging into the server

We are using a browser that access my lab's computer again. You can either use the Terminal in Rstudio server, or you can directly SSH into the machie via command line. Note that more features will be available if you 1) install PAUP* on your own machine (https://paup.phylosolutions.com/) 2) you SSH into my machine through a command line using an SSH client (native in Mac & Linux, https://www.putty.org/ or equivalent in Windows). If you would prefer this option or already know something about SSH, you can do so from the terminal using the same IP address we use for Rstudio server (this will only work if you are the VT network or VPN).

Get the data for the lab

We can update our existing git repository to the most recent version by navigating into the macrophy_course directory:

```
cd macrophy_course
git pull
```

Now navigate to the lab folder (NB: you can hit tab to autocomplete):

```
cd projects/PAUP_lab/
ls
```

We're going to make a directory for your results. Make the folder your username

```
mkdir username
```

Getting started with PAUP*

Today, we are working with data that has been simulated on a bear phylogeny in R using the following script and the program `seq-gen`. Let's look at the phylogeny itself and how it is stored.

```
more bears.tre
```

What are the numbers? How is the topology represented?

Now let's look at the data, which is saved in *NEXUS* format.

```
more bears_JC69.nex
```

To launch paup, simply type the name of the program. Because I installed the software in the system path, it can be launched from anywhere. Otherwise, you'd have to be in the directory with the program.

```
paup4.0
```

You should be greated by the following screen:

```
P A U P *
Version 4.0a (build 168) for Unix/Linux (built on Aug  2 2020 at 07:23:11)
Mon Sep 14 17:43:24 2020

        -----------------------------NOTICE-----------------------------
          This is a test version that is still changing rapidly.
          Please report bugs to dave@phylosolutions.com
        ----------------------------------------------------------------

Running on Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
    44 CPU cores on 2 sockets (hyperthreaded to 88 logical cores)
    Executable built for Intel(R) 64 architecture
    Compiled using GNU C compiler (gcc) 4.8.4
    SSE vectorization enabled
    SSSE3 instructions supported
    Multithreading enabled using Pthreads
paup>
```

Type "?" to get a list of possible commands. Type a command followed by a question mark (e.g. "lset ?") to get help about a specific function. To exit, type `quit`.

Let's begin by creating a log file of our session. Replace username with the name of the folder you created to store your results.

```
log file=./username/paup_lab.log
```

Next we execute our data block and do an exhaustive parsimony analysis.

```
execute bears_JC69.nex
outgroup Canis_lupus
```

Now conduct an exhaustive parsimony search.

```
set criterion=parsimony
alltrees
```

The best tree (or trees) are automatically saved to memory, how many trees were found? Use *ShowTrees*, *DescribeTrees* and *pscores* to examine the phylogenies (e.g. *ShowTrees 5* will examine the 5th tree saved in memory, if it exists).

We can also do a branch and bound search.

```
bandb
```

Was it any faster?

After we have found trees, we can save our results. You are going to write these results to the output folder you created earlier with your username on it.

```
savetrees file=./username/descriptive-file-name.tre format=newick root=yes brlens=yes supportvalues=node
```

Now we can estimate the tree using likelihood. We have to specify the model using the function *lset*. Use *lset ?* to see the current (default) settings.

```
set criterion=likelihood
lset ?
```

Let's start with a Jukes-Cantor model (what this data were simulated under).

```
lset nst=1 basefreq=equal
hsearch
```

How many trees were found? How do they compare to those found under parsimony? Instead of using *pscores*, use *lscores* to see the likelihood of the data given the ML tree. Also use *describetrees /plot=phylogram* to plot the tree with branch lengths. Try modifying the model lset (increasing *nst=2* or *nst=6* w/*tratio=estimate* or *rmatrix=estimate* (respectively); or setting *basefreq=empirical*) and rerun hsearch. Does this increase the likelihood (decrease the -lnL) substantially?

Now let's consider another dataset. Let's first load the modified "true tree" that I used to simulate the data. How is it different? What is the biological meaning of the changes?

```
execute bears_LBAtree.nex
describetrees /plot=phylogram
```

We are going to now load data simulated on this phylogeny, again under a JC69 model.

```
execute bears_LBA500.nex
```

Now perform a parsimony anaylsis. Switch your criterion back to parsimony and conduct branch and bound searches. How has the resulting tree changed? How does it compare to the true tree?

Let's now determine how confident we are in the tree. We're going to use a method called the boostrap, which will be explained in class.

```
bootstrap nreps=100
```

Sequentially load all the files *bears_LBAXXX.nex* which have sequences of different length and perform a bootstrap analysis. What happens as you increase the length of the sequences?

Now analyze these sequences using likelihood. Notice that the bootstrap is going to take WAAAY too long for us if we do the larger sequences. So instead of using the better search algorithms, let's make it simpler. These are quick and dirty ways of analyzing data. If you wanted publication quality, you may want to search more exhaustively.

```
bootstrap nreps=100 search=heuristic/swap=NNI nreps=1
```

What is the consequence of adding more data in the likelihood example?

Assignment: Finally... a real dataset!

We are now going to actually analyze a real dataset. We have a real sequence dataset from these species. But first, we need to establish what the correct model of sequence evolution is. We could do this all within PAUP*. In fact, if you run the following command:

```
execute bears_irbp.nex
execute modelblockPAUPb10.nex
```

It will fit a large set of models which can then be ranked. However, the output is a bit dense to parse, and the software used to summarize them is a bit hard to get these days. Folks usually select models using stand alone software like *jModeltest2*. We're going to use *modeltest-ng*, which can be run from command line. Open up another terminal tab by clicking ont he down arrow by `Terminal` and selecting `New Terminal`. Navigate to the PAUP_lab directory again, and run the following command:

```
modeltest-ng -i bears_irbp.phy
```

Note that we have two files with our sequence data, *bears_irbp.phy*, which is in Phylip format, while PAUP* will want Nexus format (*bears_irbp.nex*). There are lots of conversion tools available, I've done it already to save time.

What is the best model? Does the answer differ depending on the criterion you pick? What is the correct criterion anyway!?!

Assignment: Create a new log file with your *username_assignment2_paupanalysis.log* and implement the model in PAUP* and analyze using a heuristic search. Perform a bootstrap analysis with 100 replicates that takes a reasonable amount of time. Save the tree in your output folder.

When you are finished, upload your log file to canvas assignments.