Feedback — Phylogenetic Analysis using Parsimony

Help Center

You submitted this quiz on **Sun 14 Jul 2013 11:48 PM PDT**. You got a score of **17.00** out of **17.00**.

Overview

Note: recall that the exercises should be run from inside the virtual machine. Start that, open a web browser, go to this page and get started.

In this exercise you are going to examine three different data sets using the PAUP* program. All analyses will be performed under the parsimony criterion. Data set 1 consists of genomic nucleotide sequences from 9 primate species. This set is sufficiently small that an exhaustive search of all possible trees can be performed. Data set 2 consists of mitochondrial nucleotide sequences from 12 primates. This data set is so large that it would take too long to run through all possible trees (at least for the purpose of this exercise), but it is still small enough that branch and bound can be used to perform an exhaustive search. The third and last data set consists of Hepatitis C virus sequences isolated from different patients. This set is much too big for exhaustive searching and we will have to employ heuristic methods to analyze it.

In addition to exploring aspects of parsimonious phylogenetic reconstruction, an important goal of this exercise is to introduce you to the `PAUP*` interface, and to the different types of manipulations and analyses that can be performed within the program. Later in the course you will use PAUP* for distance-based and maximum likelihood-based phylogenetic reconstruction. Several other programs (e.g., MacClade and MrBayes) use command-line interfaces that are very similar to the one used by `PAUP*`.

Install software

```
sudo apt-get install figtree
```

```
sudo apt-get install icedtea-plugin
```

Reply "y" (followed by return) when asked whether you want to install the software. These commands install the treeviewer FigTree, and a Java-plugin for Firefox. Briefly, `sudo` allows you to run commands with administrator privileges (needed for installing software), `apt-get` is the command used to interface with the "Advanced Packaging Tool", `install` is the command given to `apt-get` to make it fetch and install a package, and `figtree` and `icedtea-plugin` are the names of the software packages we are installing.

NOTE: You need to quit and re-start the webbrowser for the plugin to be discovered. Do that now, and then continue with the exercise below.

Getting started

1. **Start Terminal window**

   Make sure to maximize the window: the analyses we will perform give lots of output to the screen, so having a nice and large shell window makes it easier to keep track of what happens.

2. **Construct working directory, copy files:**

   ```
   mkdir parsimony
   ```

   ```
   cd parsimony
   ```

   ```
   cp /home/student/data/mhc.fasta mhc.fasta
   ```

   ```
   cp /home/student/data/primate_mtDNA_interleaved.fasta primate_mtDNA_interleaved.fasta
   ```

   ```
   cp /home/student/data/hcv.fasta hcv.fasta
   ```

   ```
   ls -l
   ```

   The commands above first construct a directory called "parsimony", then changes to that directory, and finally copies three files from your data directory to the current folder (parsimony).

3. **Have a look at the data files:**

```
nedit mhc.fasta
```

This file is in the so-called fasta-format, and contains *unaligned* DNA sequences. Specifically, the sequences are major histo-compatibility complex (MHC) class I genes from nine different primate species. MHC class I genes encode proteins that are involved in the immune response. Now, close the `nedit` window and have a look at the next data file:

```
nedit primate_mtDNA_interleaved.fasta
```

This file contains mitochondrial DNA sequences from 12 different primate species. Close the `nedit` window, and have a look at the final data file:

```
nedit hcv.fasta
```

This file contains Hepatitis C virus (HCV) sequences isolated from 4 different patients. The sequenced region corresponds to the end of the E1 gene and the beginning of the E2 gene, surrounding the so-called hyper-variable region 1 (HVR1). When you've had a look close the `nedit` window so it doesn't clutter your screen.

Question 1

Multiple alignment of MHC class I sequences

In this part of the exercise, we will use the program `mafft` to make a multiple alignment of the MHC class I sequences.

- **Make multiple alignment using mafft server on EBI website:**

  Open the mafft alignment server at EBI and transfer the data from the `mhc.fasta` file (either via the text box or via the file "Browse" button). Set the sequence type to "Nucleic acid" and start the alignment by clicking "Submit".

Note 1: The starting point of all phylogenetic analyses is a multiple alignment, and your results will be no better than the quality of that initial alignment.

Note 2: There are many different multiple alignment programs. EBI provides access to some of these on its Multiple Sequence Alignment page.

- **Inspect the alignment in graphical viewer:**

  When the alignment is done: inspect the alignment by clicking the "Result summary" tab, and then the "Start Jalview" button.

- **Question:** There is a gap in the "yellow_baboon" sequence. What is the length of this gap? (number of nucleotides)

You entered:

6

Your Answer		Score	Explanation
6	✓	1.00	
Total		1.00 / 1.00	

Question 2

- **Question:** Why does this length make evolutionary sense?

Your Answer	Score	Explanation
○ The length corresponds to six amino acids and will not cause a frame shift in the reading frame.		
○ The length has no evolutionary meaning.		

○ The length corresponds to a deletion that disturbs the reading frame.		
[✓ The length corresponds to two amino acids and will not cause a frame shift in the reading frame.]	✓	1.00
○ The length corresponds to a deletion that will have a lethal outcome for the organism.		
Total		1.00 / 1.00

Question Explanation

Insertions or deletions that are not a multiplum of 3 can also occur but since they destroy the reading frame, they will often be lethal and therefore not observed.

Question 3

- **Convert alignment format to NEXUS:**

The program we will use next does not understand FASTA format, so we will first have to convert the alignment to a format it does understand, namely the so-called NEXUS format.

Close the JalView alignment viewer. On the EBI mafft "Result summary" page click the link under "Alignments in FASTA format". Select everything on the page (make sure to include everything - including the first larger than sign), and then copy and paste the alignment data to the Readseq - biosequence conversion tool at EBI. Select "PAUP|NEXUS" under "Output format" and run the converter by clicking the "Submit" button.

- **Save the alignment:**

Now, right-click the "Download" button, select "Save Link as" and save the converted file under the name `mhc.nexus` in today's working directory (parsimony - make sure you navigate to this directory so you can find the file afterwards. Also make sure to name the file mhc.nexus, and not, for instance mhc.nexus.txt).

- **Manually edit the header of the NEXUS file:**

```
nedit mhc.nexus
```

Near the top of the file find this "`MISSING=-`" and change it into "`GAP=-`". (We want to make sure the PAUP program can recognize the gaps in the input data. It is surprisingly often the case that different programs will disagree slightly on sequence file formats...). Make sure to save the file before quitting `nedit`.

Phylogenetic Analysis of MHC Class I sequences

- **Start the PAUP* program:**

  ```
  paup
  ```

  This will give you a prompt ("`paup>`") where you can write commands to the program.

- **Load the nexus file:**

  ```
  execute mhc.nexus
  ```

  This command loads the sequences from the nexus file into memory. If the nexus file had contained any commands, then these would also have been executed.

- **Exclude all alignment columns containing any gaps:**

  ```
  exclude gapped
  ```

  The `exclude` command can be used to ignore specified columns of the alignment. For instance, "`exclude 1-29 560-577`" makes PAUP* ignore columns 1 through 29 as well as columns 560 through 577. The special word "gapped" automatically selects all columns containing one or more gaps.

- **Question:** How many columns were excluded because of gaps?

You entered:

```
30
```

Your Answer		Score	Explanation
30	✓	1.00	
Total		1.00 / 1.00	

Question 4

- **Define the outgroup:**

```
outgroup olive_baboon macaque yellow_baboon
```

This defines an outgroup consisting of the three old world monkeys in the data set. The names are those that appear in the mhc.nexus file.

The outgroup will be used to place the root of the tree. The rationale is as follows: our data set consists of sequences from man, from a number of great apes, and from a number of old world monkeys. We know from other evidence that the lineage leading to monkeys branched off before any of the remaining organisms diverged from each other. The root of the tree connecting the organisms investigated here, must therefore be located between the monkeys (the "outgroup") and the rest (the "ingroup")). This way of finding a root is called "outgroup rooting".

- **Activate outgroup rooting and select how tree will be printed:**

```
set root=outgroup outroot=monophyl
```

This makes PAUP* use the outgroup we defined above for the purpose of rooting the tree. The "`outroot=monophyl`" command makes PAUP construct a tree where the outgroup is a monophyletic sister group to the ingroup. (Outroot could also have been set to "polytomy" or "paraphyl").

- **Find the most parsimonious tree by examining all possible trees:**

```
alltrees fd=barChart
```

This makes PAUP* find the best trees by exhaustively searching through all possible trees (the number of unrooted trees with 9 taxons is 135,135 - yes that is an actual number!). By default PAUP* uses the parsimony criterion for constructing trees. For each possible tree PAUP* finds the length (i.e., the number of mutational events required to explain the data set), and upon finishing this, the best tree is the one with the smallest total length. If there are several trees with the same length, then these are all kept since they are equally good. At the end of the run, PAUP* outputs a barchart giving the frequency distribution of all tree lengths. Above the histogram is a textual summary of what PAUP did and the result. Look through all of this to answer the next questions.

- **Question:** What is the length of the shortest trees?

You entered:

268

Your Answer		Score	Explanation
268	✓	1.00	
Total		1.00 / 1.00	

Question 5

- **Question:** How many trees with this score was found?

You entered:

2

Your Answer		Score	Explanation
2	✓	1.00	
Total		1.00 / 1.00	

Question 6

- **Question:** How far are the best trees from the second best ones?

You entered:

1

Your Answer		Score	Explanation
1	✓	1.00	
Total		1.00 / 1.00	

Question 7

- **Examine the best trees**

```
pscores
```

This prints the length of the trees that are currently in memory. (Actually this command not only prints but computes the lengths, so it can be used to evaluate any tree that has been loaded into memory - also trees built by other methods or other programs).

```
describetrees all/plot=cladogram
```

This will print descriptions and plots of both trees (The command "`describetrees 1`" would have printed a description of only the first tree). A cladogram is a type of tree where branch lengths are ignored and only branching order is indicated. You can scroll back to compare the two trees. You may notice that the disagreement between the two trees concerns whether gibbon or orangutan is closer to the root. Based on the information in this data set we can not distinguish between these two possibilities.

- **Question:** Note where the human sequence is located: Who are our closest relatives according to this tree? (Check all that apply).

Your Answer		Score	Explanation
☑ Chimpanzee	✓	0.20	
☑ Bonobo	✓	0.20	
☐ Gorilla	✓	0.20	
☐ Gibbon	✓	0.20	
☐ Orangutan	✓	0.20	
Total		1.00 / 1.00	

Question 8

```
describetrees all/plot=phylogram
```

A phylogram is a tree where branches are drawn with lengths proportional to the number of mutational events that has happened on them. (Note that tree-terminology is not entirely consistent, and there are several other names for this type of plot). Branch lengths are based on the reconstructed location of mutational events (and therefore also on the reconstructed ancestral sequences). Under the parsimony criterion: If a mutational event could have occurred on any one of a number of branches (in the sense that all these reconstructions give the same tree length), then each of the branches are assigned a fraction of a mutation. For instance, if a mutational event could have been placed on either of two branches, then both of them will be counted as having had 0.5 mutational events.

- **Question:** What is the longest terminal branch on the tree? (Terminal branch = branch leading to a leaf).

Your Answer		Score	Explanation
Orangutan			
Human			
Bonobo			
Gibbon	✓	1.00	
Gorilla			
Chimpanzee			
Total		1.00 / 1.00	

Question 9

```
showtrees all
```

This gives just the cladograms without further descriptions.

- **Save the trees to a file:**

```
savetrees file=mhcalltrees.nexus brlens=yes
```

This saves the trees to a file named "mhcalltrees.nexus" with indication of the location of the root, and with information about branch lengths.

- **View trees in FigTree viewer:**

Now, open a second Terminal window, change to the working directory and open the tree file with the figtree viewer as follows:

```
cd parsimony
```

```
figtree mhcalltrees.nexus
```

With this graphical viewer you can investigate the tree more closely and prepare publication grade figures. The program has several options for altering the display of the tree, including viewing the tree as unrooted, and altering the rooting interactively. After you've played around with the possibilities for a while you should close the window again.

If, later in this course, you want to construct a tree figure that is prettier than the ASCII rendition that PAUP gives you, then you need to repeat the steps performed above (save tree using the `savetrees` command, and subsequently open in FigTree). The FigTree viewer is available for several platforms.

- **Find the most parsimonious trees using branch and bound:**

Now return to the shell window where you have PAUP running and perform a branch and bound search for the best trees:

```
bandb
```

As explained in the lecture branch and bound is guaranteed to find the best trees without necessarily searching through all of tree-space.

There is therefore actually no reason to use `alltrees` unless you are explicitly interested in examining suboptimal trees or the distribution of all tree lengths.

- **Question:** What is the length of the best tree found using branch and bound?

You entered:

268

Your Answer		Score	Explanation
268	✓	1.00	
Total		1.00 / 1.00	

Question 10

- **Question:** Is that the same value found using exhaustive search?

Your Answer		Score	Explanation
No			
Yes	✓	1.00	
Total		1.00 / 1.00	

Question 11

- **Examine the branch and bound trees:**

```
showtrees all
```

(Remember: you also have the possibility of using FigTree as explained above).

- **Load the previously found trees into memory:**

We want to convince ourselves that the branch and bound trees really are identical to the trees found by the `alltrees` command. In order to do that we must now load the previously found trees back into memory while still retaining the branch and bound tree:

```
gettrees file=mhcalltrees.nexus mode=7
```

(Answer "yes" when asked whether you want to proceed). The `mode=7` command means that PAUP* should keep all trees that are currently in memory and append all trees from the file. We now have four trees in memory (the two found by `alltrees` and the two found by `bandb`). Now compare the four trees and convince yourself that the same two trees were found by `bandb`:

```
showtrees all
```

You can also compare the scores:

```
pscores all
```

As one final way of establishing that the trees are identical you can compute the "distance" between the four trees in memory:

```
treedist
```

This indicates how similar the trees in memory are by computing the so-called symmetric-difference metric. The output includes both a table listing all pairwise differences and a bar-chart giving the distribution of differences. Two trees with identical topology will have a distance of zero.

- **Question:** Are the two new trees the same as two previously found trees?

Your Answer		Score	Explanation
No			
Yes	✓	1.00	
Total		1.00 / 1.00	

Question 12

- **Define a constraint for tree-searching:**

```
constraints homooran (monophyly)=((human,orangutan));
```

This defines a constraint named "homooran" which requires the taxons "human" and "orangutan" to form a monophyletic group. The constraint tree was here given as simply: `((human,orangutan));` The tree is here shown using a notation where a pair of parentheses corresponds to an internal node, while a comma-separated list enclosed by a pair of parentheses indicates the subtrees that branch out from this internal node. The constraint tree shown above is a brief way of defining the full unresolved constraint tree: `(gorilla,gibbon,bonobo,chimpanzee,yellow_baboon,olive_baboon,macaque,(human,orangutan));`

- **Find the best tree that satisfies the constraint:**

```
bandb /constraints=homooran enforce=yes
```

This performs a branch and bound search for the best tree that has human and orangutan together as a monophyletic group. Note the option `enforce=yes` which ensures that the named constraint is applied.

- **Question:** Compare the length of this tree with the best tree found above. How many extra steps (mutations) are required in this tree? (The difference tells you something about how much better one hypothesis is than the other. There are tests that will tell you whether the difference is significant, but we will not get into that at this point in the course).

You entered:

```
15
```

Your Answer		Score	Explanation
15	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

When human and orangutang are forced to form a monophyletic group the best tree found by branch will be suboptimal (since they are not each other's nearest neighbors) and therefore require more mutations

Question 13

Phylogenetic Analysis of Mitochondrial DNA Sequences

- **Delete previously found trees from memory:**

```
cleartrees
```

- **Prepare data for analysis:**

We will now analyze the primate mitochondrial data set. Using what you learned above, perform the following steps:

- Align the sequences present in the file `primate_mtDNA_interleaved.fasta`
- Convert alignment to NEXUS format
- Save to parsimony directory in file named primate_mtDNA_interleaved.nexus
- Edit header so gaps are recognized correctly by PAUP
- Load data into PAUP
- Exclude gapped columns from data
- Set outgroup to consist of all the non-hominoid species: `Macaca_fuscata M_mulatta M_fascicularis M_sylvanus Saimiri_sciureus Tarsius_syrichta Lemur_catta`
- Activate outgroup rooting and select output of outgroup as monophyletic sister group to the ingroup.

- **Estimate time needed for exhaustive search:**

We now have 12 taxons in our data set, corresponding to 654,729,075 possible trees. We first want to estimate how long it would take to search through every single one of them, but we do not want to actually wait for the required amount of time. First, start the search as indicated below:

```
alltrees
```

Once a minute you will get an indication of how far the search has progressed. Both the percentage of all trees covered so far and the actual number of trees investigated, will be shown. After the program has printed its first status message, interrupt the run by pressing `CTRL-C`

- **Question:** Use the numbers to estimate how many minutes it would take to work through all 654,729,075 possible trees.

Note: This will obviously depend on your specific machine, so we will accept (almost) any reply.

You entered:

6.19

Your Answer		Score	Explanation
6.19	✓	1.00	

Total	1.00 / 1.00

Question Explanation

Example:

On our test machine: After 2 minutes, 14,069,140 trees had been examined. The time required to examine all 654,729,075 possible trees using exhaustive search will therefore be:
(654,729,075 / 14,069,140) * 2 min = 93 min

Question 14

- **Find the best trees using branch and bound:**

```
bandb
```

- **Question:** How long did it take to find the best trees by branch and bound (sec)? Compare this to the estimated time you would have had to wait for the `alltrees` command to finish. The time saved by ignoring suboptimal parts of search space can be quite impressive, but it depends on the structure of your data set.

Note: Again, This will depend on your computer, and we will accept a wide range of replies.

You entered:

0.19 sec

Your Answer		Score	Explanation
0.19 sec	✓	1.00	

Total		1.00 / 1.00	

Question Explanation

The Branch and bound method is guarantied to find the best tree (as opposed to heuristic methods). It acquires speed simply by skipping suboptimal parts of the search space.

Question 15

- **Perform a heuristic search using SPR:**

```
hsearch start=stepwise addseq=random nreps=100 rseed=98367 swap=SPR
```

This starts a heuristic search using rearrangements of the "subtree pruning and re-grafting" type (SPR). SPR is more elaborate than NNI and examines many more neighbors for each tree.

- **Question:** Did this search find the same best score as NNI?

Your Answer		Score	Explanation
Yes	✓	1.00	
No			
Total		1.00 / 1.00	

Question Explanation

The heuristic search using rearrangements of the "subtree pruning and re- grafting" type (SPR) found the same best score (355) as the heuristic search using rearrangements of the "nearest neighbor interchange" type (NNI).

Question 16

- **Perform a heuristic search using TBR:**

```
hsearch start=stepwise addseq=random nreps=100 rseed=98367 swap=TBR
```

This starts a heuristic search using rearrangements of the "tree bisection and reconnection" type (TBR). TBR is more elaborate than both NNI and SPR. TBR is the default swap mode for heuristic searching in PAUP*, and NNI or SPR should mostly be used if you are interested in reducing search time.

- **Question:** How many tree islands are there with the best score when using TBR?

Your Answer		Score	Explanation
1	✓	1.00	
5			
4			
3			
2			
Total		1.00 / 1.00	

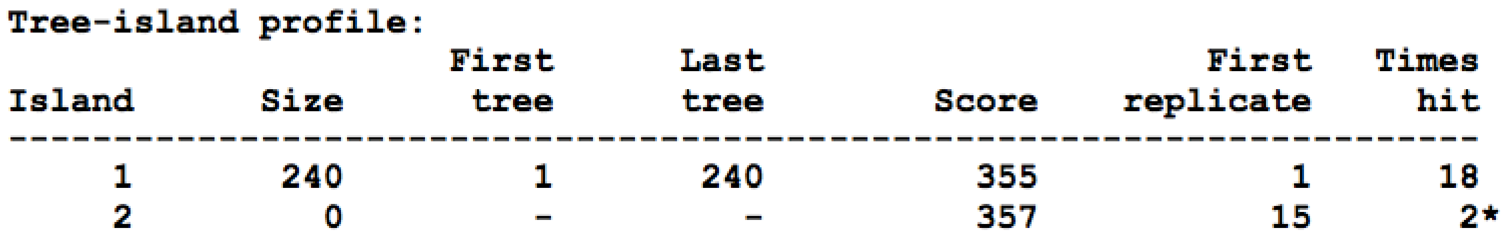
Question 17

- **Question:** Why do you think there are fewer islands with TBR and SPR compared to NNI?

Your Answer		Score	Explanation
There are more ways to rearrange a tree when using NNI.			
TBR and SPR makes smaller jumps in tree space.			
[✓ For a given tree, TBR and SPR can reach a larger number of neighboring trees than NNI.]	✓	1.00	
Trees are further away in tree space using TBR and SPR.			
The number of islands are random.			
Total		1.00 / 1.00	

Question Explanation

Only one tree island with the best score:



```
Tree-island profile:
                            First        Last                          First    Times
Island         Size          tree        tree          Score      replicate      hit
--------------------------------------------------------------------------------------
     1          240             1         240            355              1       18
     2            0             -           -            357             15        2*
```

The reason that there are fewer islands with the best score with TBR and SPR compared to NNI is that separate islands with NNI can be merged into a single island with TBR and SPR. This is because there are more ways to rearrange a tree (and therefore more neighbors) when using TBR and SPR compared to NNI. This enables the algorithm to make larger "jumps" in tree space. Trees that are further apart in tree space when using NNI (forming separate "islands") will therefore be closer when using TBR and SPR.