A Short Course in Comparative Analysis of Molecular Sequence Data.

James McInerney,
Department of Biology,
National University of Ireland Maynooth,
Co. Kildare,
Ireland.
http://bioinf.may.ie/

Index:

Requirements:

Software – RETREE (from the PHYLIP package), PAUP (www.sinauer.com).
Operating System – Can be any operating system.

A PAUP

Aim: To familiarise the user with the paup program.

PAUP is the most versatile software for comparative evolutionary analysis of DNA sequences. In this exercise, you will become familiar with how to use the software. There are hundreds of commands and options available when using the PAUP software, you will use a small subset of these commands in this practical.

To illustrate some of the commands, we shall use a dummy dataset of protein sequences.

1. Read the file 'garfield.nex' (use the unix command 'more' to read the file). The datafile is heavily-commented. Note the structure of the data file. This file is in NEXUS format, one of the most popular formats for comparative datasets. The NEXUS data format consists of a series of 'blocks'. These blocks begin with the word 'begin' and end with the word 'end;'. NEXUS files always begin with the hash mark followed by the word NEXUS.

2. Note the line that begins:

```
Begin data;
```

This line indicates that the 'data' block is to follow. This statement ends in a semi-colon. This is a standard end to a statement in the NEXUS format.

3. The next three lines indicate the size of the dataset:

```
Dimensions
    ntax=5
    nchar=21;
```

ntax means "number of taxa"
nchar means "number of characters"

NOTE 1: This statement takes the format:

```
Keyword
    option=value
    option=value;
```

NOTE 2: The statement ends in a semi-colon.

4. Information concerning the format of the dataset follows:

```
Format
    datatype=protein
    gap=-
    missing=?
    matchchar=.
    interleave;
```

Once again, the information is in the same format and the statement ends in a semi-colon.

5. The next part of the file includes a statement instructing the program to treat the INDELs in the data matrix as though they were an additional character state (a 21st amino acid)

```
    Options
        gapmode=newstate;
```

6. The next part of the file contains the sequence data. In this case, the data is protein. Note that once again this part of the file ends with a semi-colon:

```
    Matrix
        Chick     GARFIELDTHELAZY------
        Kangaroo  GARFIELDTHELAZY---CAT
        Human     GARFIELDTHE----FATCAT
        Chimp     GARFIELD-------FATCAT
        Dog       GARFIELDTHELAZYFATCAT
    ;
```

These sequences are protein and use the single-letter code for proteins. There appears to be length-variation in these sequences and therefore it has been necessary to introduce INDELs in order to construct a sensible alignment. As a result, all the sequences in this matrix are now of the same length and homologous positions have been aligned to each other. If they were not of the same length, PAUP will complain and will not read the sequences into memory. The INDELs in these sequences will be treated as though they were a 21st amino acid.

NOTE: Because of evolutionary changes, not all organisms have the same sequences. The Chick sequence, for instance is missing the FAT and the CAT domain.

7. The last part of the file contains the word:

```
    end;
```

This indicates that the data block has ended.

8. Read the data into the program's memory. This can be achieved in one of two ways. You can either specify the file name at the command line:

```
    paup garfield.nex
```

or you can start the program (type paup) and then when the program has started, you can type:

```
    execute garfield.nex
```

9. Note what has been printed to screen. Does this make sense? If not, discuss it with your demonstrator.

10. You should now be presented with the 'paup prompt':

```
    paup>
```

This indicates that you are now working within the PAUP environment.

11. The most important command within this environment is the help command. This can be used in two ways. You can either issue the help command without any trailing qualifiers:

```
    paup> help;
```

or you can type:

```
    paup> help commands;
```

which prints a one-line short description of each command. Try out both of these options now.

NOTE: It is good practice to finish each command with a semi-colon. This tells the PAUP software that you have finished issuing a command. It also means that you can issue a number of commands on one line, each one terminated by a semi-colon e.g.:

```
    paup> help; help commands;
```

12. There is a general format for commands. You can see this format if you type a question mark (?) after a command name. try this with the command for logging all output to a file.

```
    paup> log ?;
```

You should see the following:

```
Usage: Log [options...] ;

Available options:

Keyword ---- Option type ------------------------ Current default setting --
File         <log-file-name>                       garfield.log
Replace      No|Yes                               *No
Append       No|Yes                               *No
Start        No|Yes                               *No
Stop         No|Yes                               *No
FlushLog     No|Yes                                No
                                                  *Option is nonpersistent
```

The first column contains the keywords that can be used to modify the way in which the command 'log' works.

The second column contains the possible options for this keyword.

The third column contains the current setting for this option.

You can start logging your paup session to a file called 'blah.log' by issuing the command:

```
      paup> log File = blah.log start = yes;
```

Everything you type from here on and everything that appears on the screen will be simultaneously printed to a file. The format of commands is:

```
    paup> command keyword = option;
```

13. Can you figure out which command you can use to 'show the character-data matrix'? If you find the correct command, it will print the data matrix to the screen.

14. To remove the chicken sequence from the analysis type:

    paup> delete ?;

    after looking at the options, type:

    paup> delete Chick;

    you can now look at the data matrix in order to see that it has been successfully removed.

15. To restore this sequence type:

    paup> help;

    and figure out the correct command to restore the sequence.

16. To exclude the parsimony-uninformative sites type:

    paup> exclude uninf;

    look at the datamatrix now and write down the new datamatrix that results from issuing this command. Can you see why these sites are parsimony-informative?

17. Can you figure out how to include those sites you have previously excluded? write the command into your lab book.

TEST

1. Ask paup for the time. Write the exact answer in your lab book.
2. Ask paup for the current character status. Write down the output.
3. Ask paup for the current file status for the data, log and tree files.

Quit the program.

Examine the log file.

B Trees

Aim: This practical session reviews the concepts of trees and tree topologies.

You will use the program 'retree' from the PHYLIP packages of programs. The PHYLIP package was one of the first package of programs for performing phylogeny reconstruction. The package was written by Dr. Joe Felsenstein in Seattle, Washington.

Retree can be used to read a file containing the 'nested parentheses' description of a tree.
The following is an example of a nested parenthesis tree file:

```
(Cow,((Mouse,Rat),(Chimp,Human)));
```

This tree indicates that the human and chimp are each others closest relatives, the Mouse and Rat are each others closest relatives and the Cow is not specifically related to any of the terminal taxa.

Task 1: Read the treefile into memory.

1. Start the program retree by typing its name:

```
linux$ retree
```

2. You should see a screen that looks something like this:

```
-------------------------------------------------------------------------------
Tree Rearrangement, version 3.6a2.1

Settings for this run:
  U          Initial tree (arbitrary, user, specify)?  User tree from tree file
  N   Format to write out trees (PHYLIP, Nexus, XML)?  PHYLIP
  0                     Graphics type (IBM PC, ANSI)?  (none)
  W       Width of terminal screen, of plotting area?  80, 80
  L                        Number of lines on screen?  24

Are these settings correct? (type Y or the letter for one to change)
-------------------------------------------------------------------------------
```

3. We shall accept all the default options. You can do this by typing the letter 'y'. If you wanted to change any of the options, you could do so by typing the character that appears in the leftmost column (L,N,0,W,L).

4. The program now asks you for the name of the tree file. We have placed a treefile in this directory. Its name is Mammal.tre and it contains the tree detailed above. You should now be seeing the following:

```
-------------------------------------------------------------------------------
Reading tree file ...

retree: can't find input tree file "intree"
Please enter a new file name>
-------------------------------------------------------------------------------
```

Enter the name of the treefile (Mammal.tre)

5. You should now see an ASCII-character tree and a series of options like this:

```
--------------------------------------------------------------------------------
  ,>>>>>>>>>>>1:Cow
  !
--6         ,>>2:Mouse
  !  ,>>>>>8
  !  !      `>>3:Rat
  `>>7
     !      ,>>4:Chimp
     `>>>>>9
            `>>5:Human


NEXT? (Options: R . U W O T F D B N H J K L C + ? X Q) (? for Help)
--------------------------------------------------------------------------------
```

6. The program has read the nested parentheses file and has represented this tree on the screen. You can see from this treefile that the Chimp and Human are joined to each other through a single node (node number 9). Can you describe the rest of the tree and the nodes that join various groups?

7. In order to see the options that are available to us, you can type the question mark. You should then see the following:

```
--------------------------------------------------------------------------------
 . Redisplay the same tree again
 U Undo the most recent change in the tree
 W Write tree to a file
 + Read next tree from file (may blow up if none is there)

 R Rearrange a tree by moving a node or group
 O select an Outgroup for the tree
 T Transpose immediate branches at a node
 F Flip (rotate) subtree at a node
 D Delete or restore nodes
 B Change or specify the length of a branch
 N Change or specify the name(s) of tip(s)

 H Move viewing window to the left
 J Move viewing window downward
 K Move viewing window upward
 L Move viewing window to the right
 C show only one Clade (subtree) (might be useful if tree is too big)
 ? Help (this screen)
 Q (Quit) Exit from program
 X Exit from program

 TO CONTINUE, PRESS ON THE Return OR Enter KEY
--------------------------------------------------------------------------------
```

8. Type 'F' and flip the branches at node 7. Draw the resulting tree and comment on whether or not it has the same meaning as the original tree.

9. Type 'F' again and this time flip node 8. Draw the tree again and once again comment on what it now means.

10. Specify the mouse as the outgroup of the tree. How does this affect the tree? Does it make sense?

11. Experiment with options R, T, D, C and N.

12. Quit when you are finished.

C Characters

Aim: Tracing characters on phylogenetic trees.

In this practical we shall look at how different characters require different numbers of steps on different trees. We shall use the program PAUP*4.0. This program is being developed by Dr. David Swofford, originally at the Smithsonian Institute, Washington DC and now at Florida State University, Talahassee, Florida.

The dataset we shall use is a vertebrate morphological dataset. We shall use only four animals in order to show that characters do not always agree with one another. In the end, we shall use one of Charles Darwin's dictums "...the aggregate of characters" for deciding which tree topology (branching order) is the preferred one.

In this folder, you will find a dataset in "NEXUS" format. This file has a number of parts that are all equally relevant. The first line of the file begins with:

```
#NEXUS
```

This tells the program that the file is in NEXUS-format. A file in this format must conform to a very strict set of guidlines. Specifically, the file must be in discrete "blocks" of information.

The next thing we encounter in the file is a short commentary. In the same way as comments are entered into programming code, we can enter some information in an input file:

```
[!Data from: Lizard, Dog, Human and Frog                                    ]
[!This practical shows how data sometimes contain homoplastic characters    ]
[!Data are not always completely congruent                                  ]
[!However, the aggregate of characters is usually used to infer relationships ]
```

The square brackets are used to indicate a comment, the exclamation mark (!) indicates that this comment will be printed to the screen by the PAUP program.

The next part of the file is the taxa block:

```
begin taxa;
        dimensions ntax=4;
        taxlabels
                Lizard Dog Human Frog
        ;
end;
```

This block indicates that there are 4 taxa and it also indicates their names. Like all NEXUS blocks, the block begins with the word 'begin' and ends with the word 'end'. All statements end in a semi-colon.

The next block is the 'characters' block.

```
begin characters;
        dimensions
            nchar=6
            ;
        format
            symbols = "01"
            ;
```

```
        charlabels
            AMNION
            HAIR
            LACTATION
            TAIL
            ONE_JAWBONE
            PLACENTA
            ;
[!
         The first character is the Amnion
         The second character is hair
         The third character is lactation
         The fourth character is tail
         The fifth character is single bone in the lower jaw
         The sixth character is the placenta
]

matrix
        Lizard 0 0 0 1 0 0
        Dog    1 1 1 1 1 1
        Human  1 1 1 0 1 1
        Frog   0 0 0 0 0 0
        ;
end;
```

Again, the block begins with the word 'begin' and ends with the word 'end'. There is a short commentary, telling the user what each of the characters mean. Then we see the word 'matrix' which indicates that the scientific data is about to follow. The data is arranged with all homologous characters present in the same column.

As the commentary says, the first column represents the observations regarding the amnion. As we can see, this character is absent (indicated by a zero) in the Lizard and the Frog. It is present in the Human and Dog. Can you figure out the distribution of character states among the rest of the characters?

The last part of the file contains three alternative tree topologies:

```
begin trees;
        tree best   = [&U] (1,((2,3),4));
        tree second  = [&U] (1,2,(3,4));
        tree worst = [&U] ((1,3),(2,4));
;
end;
```

The trees have been given three different names - 'best', 'second', 'worst'. The four numbers that are used in the nested parentheses treefiles indicate the four taxa in the order in which they are represented in the data matrix (1 = Lizard, 2 = Dog, 3 = Human, 4 = Frog). [&U] means that the trees are not rooted, they are Unrooted.

EXERCISE

1. Start the PAUP program. This can be done in two different ways. You can either type the program name followed by the NEXUS file:

```
linux$ paup Hair_tail.nex
```

or alternatively, you can start the program by typing paup and then reading the datafile into memory using the 'execute' command:

```
linux$ paup

paup> execute Hair_tail.nex;
```

2. When the PAUP program starts, you will see a 'splash page' that looks something like this:

```
-----------------------------------------------------------------------------

P A U P *
Portable version 4.0b10 for Unix
Sat Oct  5 20:05:34 2002

      ------------------------------NOTICE------------------------------
        This is a beta-test version.  Please report any crashes,
        apparent calculation errors, or other anomalous results.
        There are no restrictions on publication of results obtained
        with this version, but you should check the WWW site
        frequently for bug announcements and/or updated versions.
        See the README file on the distribution media for details.
      ------------------------------------------------------------------

paup>

-----------------------------------------------------------------------------
```

If you read a datafile into memory at the same time as starting the program, you should see a little more information:

```
-----------------------------------------------------------------------------

Processing of file "Hair_tail.nex" begins...

Data from: Lizard, Dog, Human and Frog
This practical shows how data sometimes contain homoplastic characters
Data are not always completely congruent
However, the aggeregate of characters is usually used to infer  relationships

     The first character is the Amnion
     The second character is hair
     The third character is lactation
     The fourth character is tail
     The fifth character is single bone in the lower jaw
     The sixth character is the placenta

Data matrix has 4 taxa, 6 characters
Valid character-state symbols: 01
Missing data identified by '?'

3 trees read from TREES block
   Time used = <1 sec (CPU time = 0.00 sec)

Processing of file "Hair_tail.nex" completed.

paup>

-----------------------------------------------------------------------------
```

3. Read the data into memory now.

4. The most important command for PAUP is the 'help' command. Type this command now. You should see something like the following list of available commands:

```
-------------------------------------------------------------------------------
The following commands are always available:

 !              Edit           Help           Quit
 CD             Execute        Leave          Set
 Defaults       Factory        Log            Time
 DSet           FStatus        LSet           ToNEXUS

The following commands require data from a DATA (or TAXA and CHARACTERS) or
DISTANCES block (* = requires only TAXA block):

*Agree          *DerootTrees   *LoadConstr     Reweight       SurfCheck
 AllTrees       *DescribeTrees  LScores       *RootTrees     *TaxPartition
 AncStates       DScores       *MatrixRep      SaveAssum     *TaxSet
 Assume          Exclude        MPRSets        SaveDist      *TreeDist
 BandB           Export         NJ            *SaveTrees     *TreeInfo
 BaseFreqs       ExSet         *Outgroup       ShowAnc       *TreeWts
 Bootstrap      *Filter         PairDiff      *ShowConstr    *TStatus
 CharPartition   GammaPlot      Permute        ShowDist       TypeSet
 CharSet        *GenerateTrees  PScores        ShowMatrix    *Undelete
*ClearTrees     *GetTrees       PSet           ShowCharParts  UPGMA
 Condense        HomPart        Puzzle         ShowRateSets   UserType
*Constraints     HSearch        RandTrees      ShowTaxParts   Weights
*ConTree         Include       *RateSet       *ShowTrees      Wts
 CStatus        *Ingroup        Reconstruct    ShowUserTypes  WtSet
 CType           Jackknife     *Restore       *SortTrees
*Delete          Lake           RevFilter      StarDecomp

Type "HELP COMMANDS" or "HELP CMDS" for a one-line description of each
command.

Type "<cmdname> ?" to see brief usage and current default settings.
-------------------------------------------------------------------------------
```

5. The first command we shall use is the 'showmatrix' command. Type this command now. you should see a column containing the names of the taxa and a column containing the data. If you do not see this, then the data has not been successfully read into memory. Modify the showmatrix command so that you can see the "Character Matrix Labels" and so that the width of each column in the character matrix is 5 spaces. Re-issue the command with the modifers.

HINT: type the showmatrix command followed by a question mark.

6. The next command is the 'showtrees' command. This command will print trees to the screen. This command needs to know which trees to print to the screen (by default it just prints the first one). You could type 'showtrees 1' if you wanted to see the first tree. However, in this case, we wish to see all the trees, so you should type 'showtrees all'.

NOTE: The showtrees command takes a slightly different format to other commands. Because we with to find out some information relating to a tree in memory, we must specify which tree we are interested in. As a result, the format of the command is:

Usage: ShowTrees [tree-list] [/ options...] ;

e.g.

```
showtrees 1 3-7 9 / showtaxnum=yes;
```

7. Draw each of these trees in your lab notebook. These trees are rooted using an outgroup. In reality, they are unrooted trees. They have just been drawn in this way for simplicity.

NOTE: In some circumstances, PAUP recognises the word all.

8. We would now like to see the scores each of these trees would receive using the parsimony criterion for evaluating trees. The command for printing the parsimony scores to the screen is 'pscores'. Can you figure out how to use this command?

NOTE: the 'pscores' command has the same format as the 'showtrees' command above.

If you have successfully issued the pscores command, paup will calculate the fit of each character to the tree. It will then add these scores together and give you the 'tree length'.

9. Write down the tree length for each tree.

10. We would like to see the parsimony score for each individual character. This can be achieved using the command:

```
pscores all /single=all;
```

Type this now and record the answers. In your practical book, explain why you see the results on this screen.

11. We need to find the parsimony-informative characters. Often it is useful to exclude these characters from a dataset, since they contribute equally to the tree score for all possible trees. You can use the exclude command to remove the uninformative sites. This can be achieved using the command:

```
paup> exclude uninf;
```

How many sites were removed? Why?

Questions:
1. Which is the preferred tree using the parsimony criterion?

2. How many steps are required to describe the character 'amnion' on the first tree?

3. Which character requires the most steps on the first tree?

When you are finished, you may quit the program.

D Reconstructions

Aim: To examine different characters on trees.

In this directory you will find two files, each with the same dataset and trees.

1. Examine this dataset. You will see that this dataset consists of three blocks. The first block is the Data block. The second block is the trees block and the third block is a block consisting of commands that the paup software will read and execute. There are two commands in the input files - one to print the matrix to the screen and the other to reconstruct the characters on the various trees. In the file Vert_Hair.nex the program is being instructed to reconstruct the evolution of the character "Hair" on each of the two trees. In the file Vert_Tails.nex the program is being instructed to reconstruct the evolution of the character "Tails" on each of the two trees.

2. Read the file Vert_Hair.nex into paup. Examine the results as they are printed to the screen. Write these results into your notebook.

3. Read the file Vert_Tails.nex into paup. Examine the results as they are printed to the screen. Write these results into your notebook.

The program quits automatically each time.

E Treefit

Aim: To determine the fits of characters on alternative trees.

1. Read the input file 'data.nex' using the unix command 'more'. Note that there is a data block and a trees block. The trees block contains fifteen trees. We shall use examine the fits of characters to these trees.

2. Read the data file into paup's memory. Use the showmatrix command to ensure that the dataset has been successfully read by the program.

3. Which characters are parsimony uninformative?

4. Determine the parsimony scores for each tree. You can find the options by typing the command followed by a question mark.

```
paup> pscores ?;
```

*Note that this command has two discrete parts. Immediately following the command, you must supply a tree or number of trees, so that paup can return the figures that are relevant for these trees. If you want to modify the default output for the options within the 'pscores' command, you must first use a slash (/). In this case, you might issue a command like:

```
paup> pscores 1 /total=no;
```

or

```
paup> pscores 5 /total=yes;
```

try these options now. You can find out the parsimony scores for all trees using the command:

```
paup> pscores all;
```

5. Exclude the parsimony-uninformative scores and determine the scores again. Why are there differences?

6. Which tree is the most parsimonious?

7. Using the 'pscores' command, find out the character statistics for every 'single' character for tree number 1.

8. Reconstruct the evolution of character 8 on tree number 11. Draw the answer in your lab book. *Note that the reconstruct command also has an unusual syntax.

Quit the program.

F Heuristic searches.

Aim: To search alternative trees using heuristic methods.

When the number of sequences to be evaluated is larger than about 10, it is necessary to use approximate methods in order to evaluate alternative tree topologies. These methods are called heuristic searches. There are a wide variety of tree searches and we will use only a few of these. The usual approach is to generate a tree (in some fast way) and then swap branches on this tree, looking for trees with better scores.

The dataset we shall use is a set of primate mitochondrial genes. This dataset was collected in order to solve the age-old question concerning the relationship between humans, chimpanzees and gorillas. These are the real data used in the publication in 1988.

1. Take a look at the dataset. Its format is slightly different, as we are using a dot "." to indicate that a nucleotide has the same character state as the first sequence in the dataset (the lemur). There is also a numbering system on top of the data matrix that indicates the position in the alignment.

2. Read the dataset into the program's memory.

3. Look at the options for a heuristic search (command 'hs'). The first of the options concerns the kind of heuristic search that can be carried out. The choice is between a "Nearest Neighbor Interchange", a "Sub-tree Pruning and Regrafting" or a "Tree-bisection and Reconnection" search. These three kinds of search are in order of the rigorousness of the approach.

4. Perform a search of trees using the NNI procedure, record the number of trees searched and the score of the best tree.

5. Use the showtrees command to see the best tree(s).

6. Perform a search of trees using the SPR procedure, record the number of trees searched and the score of the best tree.

7. Use the showtrees command to see the best tree(s).

8. Perform a search of trees using the TBR procedure, record the number of trees searched and the score of the best tree.

9. Use the showtrees command to see the best tree(s).

10. Write in your copybook your interpretation of these trees. What does this mean for the relationships between these primate species?

11. Use the 'contree' command to generate a consensus tree. What part of the tree is 'collapsed' into a trichotomy? Why?

12. Quit the program.

G Exhaustive Searches

Aim: To search through all possible tree topologies in order to find the most parsimonius.

In this practical, we shall examine all possible tree topologies for a collection of sequences. Exhaustive searching of all tree topologies is by far the best method of finding the most parsimonious solutions. However, for very large datasets, it is impractical. The number of possible tree topologies is:

$$T = (2n-5!)/(2n-3(n-3)!)$$

The growth in tree numbers proceeds as follows:

n	T
4	3
5	15
6	105
7	945
8	10,395
9	135,135
10	2,027,025

However, for small numbers of sequences, we can search through all tree topologies relatively easily.

1. Take a look at the data file called BigHIV.nex. This file contains a total of eight sequences of HIV viruses isolated from patients in a variety of different countries. Read the datafile into paup's memory.

2. Confirm that you have successfully read the file into memory by printing the matrix to the screen.

3. Take a look at the options for searching through "all trees". You can ask the program to search through all these trees by simply issuing the relevant command without any options. Do this now.

4. Note the number of trees that were evaluated and the scores of the shortest and the longest trees that were encountered.

5. There will be a print out of the tree scores that were encountered during the search. It should look something like this:

```
================================================================================
Frequency distribution of tree scores:

    mean=24.572872 sd=2.034705 g1=-0.787999 g2=0.198534
   /------------------------------------------------------------------
16 | (5)
17 | (13)
18 |# (34)
19 |##### (151)
20 |####### (205)
21 |############### (449)
22 |################################ (978)
23 |######################### (771)
24 |########################################################################## (2056)
25 |################################################### (1595)
26 |########################################################################### (2070)
27 |########################################################################### (2068)
   \------------------------------------------------------------------
================================================================================
```

This frequency distribution indicates the number of times a tree of a particular length was encountered during a tree search. We expect that well-structured data with low levels of homoplasy and strong phylogenetic signal will produce a frequency distribution where the largest portion of trees are much longer than the most parsimonious solutions.

6. Run the exhaustive search again, this time changing the Frequency Distribution display to 'Histogram" and printing the output to a file called "BigHIV.freq". Take a look at the output file, note the number of trees.

7. Quit the program.

H Random

Aim: To perform a search of tree space using a dataset that has random sequences. We will demonstrate what "tree space" looks like for this dataset.

1. Take a look at the dataset and then read this dataset into paup's memory.

2. Perform an exhaustive search of tree space.

3. Note the distribution of tree lengths.

4. Record in your lab books how this distribution differs from the distribution in the previous exercise. This distribution is indicative of a dataset that does not contain well-structured and phylogenetically robust data.

3. Quit the program.

I Bootstrap

Aim: To evaluate the level of support for internal branches in a phylogenetic tree. We shall demonstrate the differences we might see in "good" and "poor" data sets.

We use a resampling procedure such as bootstrapping in an effort to evaluate the relative levels of support and conflict for various sets of relationships. The original data is sampled with replacement and pseudo-replicates of the original matrix are used to construct trees. This resampling strategy is repeated a large number of times and at the end the data is summarised using a majority-rule consensus procedure.

1. Read the HIV.nex dataset into paup's memory. Take a look at the bootstrap commands options. The first option is BSeed. This is a seed number that computers sometimes need in order to generate a random number. PAUP will usually use the clock time as a random number.

2. Use the 'bootstrap' command to perform a bootstrap analysis using the default parameters. Record the random 'starting seed' number. Draw the resulting tree into your lab book with the support values associated with the internal branches.

3. You will also see a 'partition table' that details the number of times a particular internal branch was seen during the resampling procedure. you must remember that during bootstrap resampling, it is likely that trees will be produced that do not look like the final consensus tree.

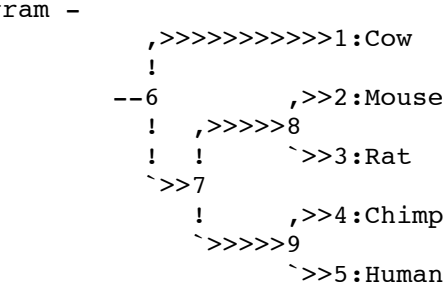
This is an example of a partition table:

12345678	Freq
.***....	79.18
..**....	76.76
.******.	72.31
.....**.	52.28
....***.	50.51
....****	23.69
.**.....	16.47
.***.**.	15.65
.*..***.	6.43

This indicates that during the analysis a branch that separated sequences 1,5,6,7 and 8 from the rest was seen 79.18% of the time. Can you work out how many times a branch was seen that separated sequences 2 and 3 from the rest? Would this branch be seen in the majority-rule consensus tree?
*NOTE:

The following notations can be used to describe the same tree:

Tree diagram -



```
Nested parentheses -
                ((Human, Chimp),(Mouse, Rat), Cow);


Partition table -
                    12345
                    ...**
                    .**..
                    .****
```

4. Read the Random.nex file into paup's memory. Perform the same bootstrap analysis on this data set. Record the results and indicate how the trees differ in terms of the degree of support seen throughout the tree. Bear in mind that the Random.nex file contains sequences that we have randomised - there is no real set of phylogenetic relationships.

5. Perform the bootstrap analyses again on both datasets, this time alter the number of replicates so that you perform 1,000 resampling iterations.

6. Record the results and quit the program.