

# Chapter 6. Phylogeny Inference Based on Maximum Likelihood Methods with **TREE – PUZZLE**

Arndt von Haeseler<sup>(1, 2)</sup> and Korbinian Strimmer<sup>(3)</sup>

<sup>(1)</sup>Mathematisch-Naturwissenschaftliche Fakultät, Heinrich Heine Universität, Düsseldorf, Germany. <sup>(2)</sup>John von Neumann-Institute for Computing, Forschungszentrum Jülich, Jülich, Germany. <sup>(3)</sup>Department of Statistics, University of Munich, Munich, Germany.\

## **THEORY**

### **6.1. Introduction**

The concept of likelihood deals with situations, which typically arise in natural sciences, where given some data  $\mathbf{D}$  a decision has to be made about an adequate explanation of the data. Thus, a specific model and a hypothesis are formulated, where the model as such is generally not in question. In the phylogenetic framework one part of the model is that the sequences actually evolve according to a tree. The possible hypotheses are the different tree-structures, the branch lengths, the parameters of the model of sequence evolution and so on. By assigning values to these elements it is possible to compute the probability of the data and to make statements about the plausibility of these values. If the hypothesis varies, it will then turn out that some hypotheses produce the data with higher probability than others. A standard example is the coin tossing instance. After flipping a coin  $n=100$  times,  $h=21$  heads and  $t=79$  tails have been observed. Thus  $\mathbf{D} = (21, 79)$  constitutes the data. The model then states that with some probability  $\theta \in [0, 1]$  head appears when the coin is flipped. Moreover, it is assumed that the outcome of each experiment is independent of the others, that  $\theta$  does not change during the experiment, and that the experiment has only two outcomes (head or tail). The model is now fully specified. Since both heads and tails were obtained,  $\theta$  must be larger than zero and smaller than 1. Moreover, any probability

textbook tells that the probability to observe exactly  $H=h$  heads in  $n$  experiments can be calculated according to the binomial distribution as follows:

$$\Pr[H = k] = \binom{n}{h} \theta^h (1-\theta)^{n-k} \quad (6.1)$$

Equation (6.1), the so-called binomial formula, can be read in two ways. First, it is assumed that  $\theta$  is known then the probability of  $h = 0, K, n$  heads in  $n$  tosses can be computed. Second, equation (6.1) can be seen as a function of  $\theta$ , where  $n$  and  $k$  are given, this defines the so-called **likelihood function**

$$L(\theta) = \Pr[H = k] = \binom{n}{h} \theta^h (1-\theta)^{n-h} \quad (6.2)$$

Figure 6.1 displays the **likelihood function** for the coin example and it can clearly be seen that some hypotheses, i.e. choices of  $\theta$ , generate the observed data with a higher probability than others. In particular, equation 6.2 becomes maximal if  $\theta = 21/100$ . This value can also be computed analytically. To ease computation, first compute the logarithm of the **likelihood function**, which leads to sums instead of products

$$\log[L(\theta)] = \log \binom{n}{h} + h \log \theta + (n-h) \log(1-\theta) \quad (6.3)$$

The problem is now to find out the value of  $\theta$  maximizing the function. From elementary calculus it is known that the maximum of a function  $y=f(x)$ , when it exists, is given by the value of  $x$  for which the first derivative of the function equals to zero. Differentiation of equation 6.3 with respect to  $\theta$  yields to:

$$\frac{\partial \log[L(\theta)]}{\partial \theta} = \frac{h}{\theta} + \frac{n-h}{1-\theta} \quad (6.4)$$

Such derivative is equal to zero if  $\theta = h/n$ , positive for smaller values of  $\theta$ , and negative for larger values, so that  $\log[L(\theta)]$  attains its maximum when  $\hat{\theta} = h/n$ . Thus  $\hat{\theta} = h/n$  is the **maximum-likelihood estimate (MLE)** of the probability of observing a head in a single coin toss. In general, when the value of  $\theta$  that maximizes equation 6.3 is selected, the observed data are produced with the highest likelihood. This is precisely the **maximum likelihood principle**. One should be aware however, that the resulting likelihoods are usually small (for example, it is already  $L(21/100) \approx 0.0975$ ). On the other hand, one can compare the likelihoods of competing hypotheses, by computing the odds-ratio. Note that the hypothesis that the coin is a fair ( $\theta = 1/2$ ), leads to a likelihood of  $L(1/2) \approx 1.61 \cdot 10^{-9}$ , thus the **MLE** of  $\theta$  is  $6 \cdot 10^7$  times more likely to produce the data! This comparison of odds-ratios leads to a statistical test procedure that is discussed in chapter 10.

In evolution, point mutations are considered chance events just like tossing a coin. Therefore, at least in principle, the probability of finding a mutation along one branch in a **phylogenetic tree** can be calculated by using the same **maximum likelihood** framework discussed in the previous section. The main idea behind phylogeny inference with **maximum likelihood** is to find out the tree topology, the branch lengths, and the parameters of the evolutionary model (transition-transversion ratio, base frequencies, rate variation among-sites, ect., see chapter 4) which maximize the probability of observing the sequences at hand. In other words, the **likelihood function** is the conditional probability of the data (sequences) given a hypothesis (a model of substitution with a set of parameters  $\theta$ , and the tree  $\tau$ , including branch lengths):

$$L(\tau, \theta) = \text{Prob}(\text{Data} \mid \tau, \theta) = \text{Prob}(\text{Aligned sequences} \mid \text{tree, model of evolution}) \quad (6.5)$$

The **maximum likelihood estimates (MLEs)** of  $\tau$  and  $\theta$  are those making the **likelihood function** as large as possible:

$$\hat{\tau}, \hat{\theta} = \max_{\tau, \theta} L(\tau, \theta) \quad (6.6)$$

Before proceeding to the next section, some cautionary notes are necessary. First, it is important to realize that the *likelihood function* must not be confused with a probability. It is defined in terms of a probability, but this probability is the probability of the observed event and not of the unknown parameters. The parameters have no probability because they do not depend on chance. Second, the probability of getting the observed data has nothing to do with the probability that the underlying model is correct. For example, if the model states that the sequences evolve according to a tree, although they have recombined, then the final result will still be some tree that gives rise to the *maximum likelihood* value (see also chapter 14). The probability of the data given the *MLE* of the parameters does not provide any hints that the model assumptions are true. One can only compare the *maximum likelihood* values with other likelihoods for model parameters that are elements of the model. If one wants to know if the hypothesis of tree-like evolution is reasonable one has to enlarge the type of relationship allowed among sequences; but this is not discussed here.

## 6.2. The formal framework

Before entering the general discussion about *maximum likelihood* tree reconstruction, the simplest example, namely reconstructing a *maximum likelihood* tree for two sequences, will be considered. A tree of two *taxa* has only one branch connecting the two sequences and the whole purpose of the exercise is the reconstruction of the branch length that produces the data with maximal probability.

### 6.2.1. The simple case: maximum likelihood tree for two sequences

It is assumed that the sequences are evolving according to the Jukes Cantor model (see Chapter 4). Each position evolves independently of the remaining sites and with the same evolutionary rate. The alignment is of length  $l$  for the two sequences  $S_i = (s_i(1), K, s_i(l)), (i = 1, 2)$ , where  $s_i(j)$

is the nucleotide, the amino acid or any other letter from a finite alphabet at sequence position  $j$  in sequence  $i$ . The **likelihood function** is then, according to equation 4.31,

$$L(d) = \prod_{j=1}^l \pi_{s_1(j)} P_{s_1(j)s_2(j)} \left( -\frac{d}{3/4} \right) \quad (6.7)$$

where  $d$ , the number of substitutions per site, is the parameter of interest and  $P_{xy}(t)$  is the probability to observe nucleotide  $y$  if nucleotide  $x$  was originally present. Following equations 4.12a and 4.12b we obtain

$$P_{xy} \left( -\frac{4}{3}d \right) = \begin{cases} \frac{1}{4} \left( 1 + 3 \exp \left[ -\frac{4}{3}d \right] \right) \equiv \tilde{P}_{xx}(d), & \text{if } x = y \\ \frac{1}{4} \left( 1 - \exp \left[ -\frac{4}{3}d \right] \right) \equiv \tilde{P}_{xy}(d), & \text{if } x \neq y \end{cases} \quad (6.8)$$

To infer  $d$  the relevant statistics is the number of identical pairs of nucleotides ( $l_0$ ) and the number of different pairs ( $l_1$ ), where  $l_0 + l_1 = l$ . Therefore the alignment is summarized as  $\mathbf{D} = (l_0, l_1)$  and compute the score of equation 6.3 as:

$$\log[L(d)] = C + l_0 \log[\tilde{P}_{xx}(d)] + l_1 \log[\tilde{P}_{xy}(d)] \quad (6.9)$$

that is maximized if

$$d = -\frac{3}{4} \log \left[ 1 - \frac{4}{3} \cdot \frac{l_1}{l_1 + l_0} \right] \quad (6.10)$$

In practice, the **MLE** of the number of substitution per site equals the method of moments estimate (Equation 4.15a). Therefore the **maximum likelihood** tree relating the sequences  $S_1$  and  $S_2$  is a straight line of length  $d$  with the sequences as endpoints.

The above example was completely computable, because it is the simplest model of sequence evolution and, more importantly, because only two sequences, which can only be related by one

tree, were considered. The following paragraphs sets up the formal framework to study more sequences.

### 6.2.2. The complex case.

When the data set consists of  $n$  ( $n > 2$ ) aligned sequences, instead of computing the probability  $P(t)$  of observing two nucleotides at a given site in two sequences, it is computed the probability of finding a certain column, or pattern of nucleotides, in the data set. Let  $D_j$  denote the nucleotide pattern of site  $j = 1, K, l$  in the alignment (see Figure 6.2.). The unknown probability obviously depends on the model of sequence evolution,  $M$ , and the tree,  $T$ , relating the  $n$  sequences together with the number of substitutions along each branch of the tree (i.e. the branch lengths). In theory one could assign each site its own model of sequence evolution according to the General Time Reversible model (Chapter 4) and its own set of branch lengths. Then, however, the goal to reconstruct a tree from an alignment gets almost computationally intractable. Therefore, several simplifications are needed. First, it is assumed that each site  $s$  in the alignment evolves according to the same model  $M$ , for example, the Tamura Nei (TN) model (Equations 4.32a, b, c), that is  $\gamma$ ,  $\kappa$ , and  $\pi$  are the same for each site in the alignment. Note that the assumption implies that all sites evolve at the same rate  $\mu$  (Equation 4.24). To overcome such a simplification, the rate at a site is modified by a rate specific factor  $\rho_j > 0$ . Thus, the ingredients for the probability of a certain site pattern are at hand and

$$\Pr[D_s, T, M, \rho_j], j = 1, K, l \quad (6.11)$$

specifies the probability to observe pattern  $D_j$ . If it is assumed also that each sequence site evolves independently, i.e. according to  $T$ ,  $M$ , with a site specific rate  $\rho_j$ , then the probability of observing the alignment (data)  $\mathbf{D} = (D_1, K, D_l)$  equals

$$L(T, M, \boldsymbol{\rho}) \equiv \Pr[\mathbf{D}, T, M, \boldsymbol{\rho}] = \prod_{j=1}^l \Pr[D_j, T, M, \rho_j] \quad (6.12)$$

When the data are fixed, equation 6.12 is again a *likelihood function* (cf. equations 6.2 and 6.5), which allows for the dual way of looking at it (cf. previous section). First, for a fixed choice of  $T$ ,  $M$ , and the site rate vector  $\boldsymbol{\rho}$ , the probability to observe the alignment  $\mathbf{D}$  can be computed with equation 6.11. Second, for a given alignment  $\mathbf{D}$  equation 6.12 can be used to find the *MLEs*.

In what follows, both issues will be treated separately. However, to make matters easier, it is assumed that the site specific rate factor  $\rho_j$  is drawn from a  $\Gamma$ -distribution with expectation 1 and variance  $1/\alpha$  (Uzzel and Corbin, 1971; Wakely 1993, where  $\alpha$  defines the shape of the distribution; see also 4.6.1.).

### 6.3. Computing the probability of an alignment for a fixed tree.

In the following the tree  $T$  with its branch lengths (number of substitutions), the model of sequence evolution  $M$  with its parameters (i.e., transition-transversion ratio, stationary base composition etc.) and the site specific rate factor  $\rho_j=1$  for each site  $j$ . The goal is to compute the probability of observing one of the  $4^n$  possible patterns in an alignment of  $n$  sequences. To illustrate the principle, set  $n=4$  and study the four sequences tree displayed in Figure 6.3.

Since the model  $M$  belongs to the GTR class, i.e. is a time reversible model (see chapter 4), it is assumed that evolution started from sequence  $S_0$  and then proceeded along the branches of the tree  $T$  with branch lengths  $(d_1, d_2, d_3, d_4, d_5)$ . To compute  $\Pr[D_j, T, M, \mathbf{1}]$  for a specific site  $j$ , where  $D_j = (s_1, s_2, s_3, s_4)$  are the nucleotides observed, it is necessary to know the ancestral states  $s_0$  and  $s_5$  (see below). Then the conditional probability of the data given the ancestral sates will be:

$$\Pr[D_j, T, M, \mathbf{1} | s_0, s_5] = P_{s_0 s_1}(d_1) \cdot P_{s_0 s_2}(d_2) \cdot P_{s_0 s_5}(d_5) \cdot P_{s_5 s_3}(d_3) \cdot P_{s_5 s_4}(d_4). \quad (6.13)$$

The computation follows immediately from the considerations in chapter 4. However, in almost any realistic situation the ancestral sequences are not available. Therefore it is necessary to sum

over all possible combinations of ancestral states of nucleotides. As discussed in chapter 4 (section 4.4), nucleotide substitution models assume *stationarity*, i.e the relative frequencies of A, C, G, and T ( $\pi_A, \pi_C, \pi_G, \pi_T$ ) are at equilibrium. Thus, the probability for nucleotide  $s_0$  will equal its stationary frequency  $\pi(s_0)$  from which it follows that

$$\Pr[D_j, T, M, \mathbf{1}] = \sum_{s_0} \sum_{s_5} \pi(s_0) P_{s_0 s_1}(d_1) \cdot P_{s_0 s_2}(d_2) \cdot P_{s_0 s_5}(d_5) \cdot P_{s_5 s_3}(d_3) \cdot P_{s_5 s_4}(d_4). \quad (6.15)$$

Although this equation looks like one needs to compute exponentially many summands, it is possible to evaluate the sum very efficiently by evaluating the likelihoods moving from the end-nodes of the tree to the root (Felsenstein 1981). In each step two nodes from the tree are removed and replaced by a single node. This process bears some similarity with the computation of the minimal number of substitution on a given tree in the maximum parsimony framework (Fitch 1971). However, contrary to *maximum parsimony* the distance (number of substitutions) between the two nodes is taken into account. If two sequences share the same nucleotide, then under *parsimony* the most recent common ancestor also carries this nucleotide (see next chapter), whereas in the *maximum likelihood* framework this nucleotide is shared by the ancestor only with a certain probability, which gets small as the sequences are only very remotely related.

### 6.3.1. Felsenstein's pruning algorithm

Equation 6.14 shows how to compute the likelihood of a tree for a given position in a sequence alignment. In order to generalize this equation for more than four sequences it is necessary to summarize over all possible assignment of nucleotides at the  $n-2$  inner nodes of the tree. Unfortunately, this straightforward computation is unfeasible, but one can reduce the amount of computation considerably by noticing the following recursive relationship in a tree. Let  $D_j = (s_1, s_2, s_3, \dots, s_n)$  be a pattern at a site  $j$  and tree  $T$  and a model  $M$  are fixed. Nucleotides at

inner nodes of the tree are abbreviated as  $x_i, i = n + 1, \dots, 2n - 2$ . For an inner node  $i$  with offspring  $o_1$  and  $o_2$  the vector  $\mathbf{L}_j^i = (L_j^i(A), L_j^i(C), L_j^i(G), L_j^i(T))$  is defined recursively as

$$L_j^i(s) = \left[ \sum_{x \in \{A, C, G, T\}} P_{s,x}(d_{o_1}) L_j^{o_1}(x) \right] \cdot \left[ \sum_{x \in \{A, C, G, T\}} P_{s,x}(d_{o_2}) L_j^{o_2}(x) \right], s \in \{A, C, G, T\} \quad (6.15)$$

with

$$L_j^i(s) = \begin{cases} 1, & \text{if } s = s_i \\ 0 & \text{otherwise,} \end{cases} \quad (6.16)$$

where  $d_{o_1}$  and  $d_{o_2}$  are the number of substitutions connecting node  $i$  and its descendants in the tree. Without loss of generality, it is assumed that the node  $2n-2$  has three offspring  $o_1, o_2$ , and  $o_3$ , respectively. For this node equation 6.15 is modified accordingly. These two equations allow a much more efficient computation of the likelihood for each position of an alignment by realizing that

$$\Pr[D_j, T, M, \mathbf{1}] = \sum_{s \in \{A, C, G, T\}} \pi_s L_j^{2n-1}(s). \quad (6.17)$$

Equation 6.17 can then be used to compute the likelihood of the full alignment with the aid of equation 6.12. In practice, one avoids the calculation of products, but moves rather to log-likelihoods. i.e. equation 6.12 turns into

$$\text{Log}[L(T, M, \mathbf{1})] = \text{Log} \left[ \prod_{j=1}^l \Pr[D_j, T, M, \mathbf{1}] \right] = \sum_{j=1}^l \text{Log}[\Pr[D_j, T, M, \mathbf{1}]]. \quad (6.18)$$

#### 6.4. Finding a *maximum likelihood tree*

Equations 6.15 – 6.18 show how to compute the likelihood of an alignment, if everything were known. In practice, however, the branch lengths of the tree are not known. The computation of the branch lengths is done numerically by maximizing equation 6.18, i.e. by finding those branch lengths for the tree  $T$  maximizing the log-likelihood function. This is accomplished by applying

Newton's method or other numerical routines. Such computation is usually very time consuming and sometimes the result depends on the numerical method. Nevertheless maximizing the likelihood for a single tree is not the major challenge in phylogenetic reconstruction. The daunting task is actually to find the tree among all possible tree structures that maximizes the global likelihood. Unfortunately, for any method that has an explicit optimality criterion (e.g., *maximum parsimony*, *distance* methods, and *maximum likelihood*) no algorithms are known that guarantee the localization of the best tree(s) in the huge space of all possible tree topologies. The naive approach to simply compute the *maximum likelihood* value for each tree topology is prohibited by the huge number of tree structure even for moderately sized data sets, which can be computed for  $n$  sequences according to

$$t_n = \frac{(2n-5)!}{2^{n-3}(n-3)!} = \prod_{i=1}^n (2i-5) \quad (6.19)$$

When computing the *maximum-likelihood* tree one has to compute for each tree the model parameters and the branch lengths and then to pick the tree that yields the highest likelihood. Testing all possible trees is impossible, due to the large number of tree topologies and it is also not computational feasible to estimate the model parameters for each tree. Thus, various heuristics are used to suggest reasonable trees. Among them are stepwise addition (for example used in Felsenstein's *PHYMLIP* package (Felsenstein, 1993), star decomposition, also the basis of neighbor joining (Saitou and Nei 1987), the *MOLPHY* package (Adachi and Hasegawa, 1996) or of *PAML* (Yang, 1997). These methods will be discussed in the next chapter, since they are all implemented in the *PAUP\** program and they can be used as well when looking for trees with *maximum parsimony* (Swofford et al., 1995). What follows will focus on the description of another heuristic to find a plausible candidate as the *maximum likelihood* tree: the *quartet puzzling* method implemented in the *TREE-PUZZLE* program (Strimmer and von Haeseler, 1996a).

#### 6.4.1 The quartet puzzling algorithm

Given a set of  $n$  aligned nucleotide (or amino acid) sequences, it is called a *quartet* any randomly chosen group of four of them. The *quartet puzzling* algorithm analyzes all the possible (or a randomly chosen sample) of *quartets* in a data set, taking advantage of the fact that for a *quartet* just three unrooted tree topologies are possible. In essence the algorithm is a three-step procedure. The first step, the so-called *maximum-likelihood step*,

computes for each of the  $\binom{n}{4}$  possible quartets the *maximum likelihood* values

$L_1 \equiv L(T_1, M, \rho), L_2 \equiv L(T_2, M, \rho), L_3 \equiv L(T_3, M, \rho)$  for the three possible four-sequence trees  $T_1, T_2, T_3$  assuming a user defined model  $M$  of sequence evolution. The resulting list of

$3 \cdot \binom{n}{4}$  likelihoods is then used in the *quartet-puzzle step* to compute an intermediate trees by

inserting sequences sequentially in an already reconstructed *subtree*. Figure 6.4 illustrates the insertion procedure, where a tree with sequences A, B, C, and D is already reconstructed and sequence E is inserted according to the four sequence trees that contain sequence E and three sequences from the partial tree. Eventually, sequence E is inserted at the branch with minimal penalty. Figure 6.5. shows the all intermediate trees for a small example of five sequences. This step is repeated at least a thousand times for various input orders of sequences to avoid reconstruction artifacts due to the ordering of the sequences and to get a representative collection of trees. Finally, in step three, the *majority-rule consensus* (Margush 1981) is computed from the resulting intermediate trees (see also chapter 5). The resulting tree is called the *quartet-puzzling tree*. Table 6.1 shows the percentage of clusters found in the five sequence example.

The consensus step provides information about the number of times a particular grouping occurred in the intermediate trees. This so-called *reliability-value* or *support-value*, measures (in %) how frequently a group of sequences occurs among all intermediate trees. All groups that

occur in more than 50% of the collection of intermediate trees are represented in the majority rule *consensus tree*. One should be aware that this *consensus tree* is not necessarily the *maximum likelihood* tree. In the toy example in Table 6.1, the TREE-PUZZLE program would output the tree grouping sequences A and E (support value 72%) and C and D (support value 52%). In this case the tree is fully resolved. If the data do not provide a good resolution of phylogenetic relationships for a subgroup of sequences, the *consensus tree* will display a small *reliability value* for the corresponding internal branch of the tree. Note that the *reliability value* should not be confused with the usual *bootstrap values*. Whereas *reliability values* are an intrinsic result of the *quartet-puzzling* algorithm, *bootstrapping* is an external procedure that can be applied to any tree building method (cf. chapter 5).

### 6.5. Estimating the model parameters with *maximum likelihood*

When inferring parameters of a model that allows for substitution rate heterogeneity across sites (cf. section 4.10.1) it is necessary to consider an overall tree because the relation of the sequences is important for the estimation (SULLIVAN *et al.* 1996). In theory it is possible to optimize both the parameters of the model and the tree topology with *maximum likelihood* by simultaneously finding the *phylogenetic tree* and the set of parameters maximizing the likelihood function. However, for large data sets (more than 10-15 sequences) such approach becomes computationally unfeasible. Therefore, the accepted strategy is to infer a “reasonable” tree topology with faster though less reliable reconstruction methods and use that tree to estimate the parameters. Eventually a *maximum likelihood* tree can be re-estimated with the new set of parameters. This approach assumes that parameters estimates are not heavily disturbed when using a slightly incorrect topology. Among the fast distance based tree reconstruction methods *neighbor-joining* (Saitou and Nei 1987) exhibits the best performance in terms of accuracy and speed. Though its probability to find the true underlying tree decreases rather quickly with the

number of *taxa* it has been shown that the *neighbor-joining* tree is always very similar to the true tree even for a large number of sequences (Strimmer and von Haeseler 1996b).

The following iterative procedure is implemented the in TREE-PUZZLE program:

- i. Based on reasonable pair-wise *genetic distance* estimates a neighbor-joining tree is computed.
- ii. Then *maximum likelihood* branch lengths are computed for this tree topology and parameters of sequence evolution are estimated.
- ii. Based on these estimates a new *neighbor-joining* tree is computed and procedure (2) is repeated.

Steps (i) and (ii) are performed until the estimate of the model parameters are stable. Eventually, given the unrooted tree topology and a set of optimized parameters, *maximum likelihood* branch lengths can be computed using the *likelihood function* (equation 6.12), which involves simultaneous optimization of  $2n-3$  branch lengths for a tree with  $n$  sequences. The computing time can be accelerated by estimating an approximate likelihood. Adachi and Hasegawa (1996) have applied a least-squares fit of pair-wise *maximum likelihood* distances to a given tree topology and have shown that the resulting sub-optimal set of branch length estimates is close to the *maximum likelihood*. TREE-PUZZLE employs this idea to obtain approximate estimates of model parameters, saving computation time and still serving as an efficient tool to estimate model parameters.

## PRACTICE

### 6.6. Software packages

A large number of software packages to compute *maximum likelihood* trees from DNA or amino acid sequences exist. A detailed list can be found at Joe Felsenstein's web page

<http://evolution.genetics.washington.edu/phylip/software.html>. Since program packages are emerging at a rapid pace the reader is advised to look up this web page in regular intervals to be up to date. The reader would also like to point to the software packages mentioned in chapter 4 and their description given.

## 6.7. An illustrative example

In what follows `hivALN.phy` file will be analyzed with the latest version of TREE-PUZZLE (5.0). Place the `hivALN.phy` file in the TREE-PUZZLE folder and run the executable, the following text appears:

```
WELCOME TO TREE-PUZZLE 5.0.pl6!
```

```
Please enter a file name for the sequence data:
```

```
enter the filename: hivALN.phy and press return:
```

```
Input data set contains 14 sequences of length 2781
```

```
(consists very likely of nucleotides)
```

```
GENERAL OPTIONS
```

```
b           Type of analysis?  Tree reconstruction
k           Tree search procedure?  User defined trees
z    Compute clocklike branch lengths?  No
e           Parameter estimates?  Approximate (faster)
x           Parameter estimation uses?  1st input tree
```

```
SUBSTITUTION PROCESS
```

```
d           Type of sequence input data?  Auto: Nucleotides
h           Codon positions selected?  Use all positions
```

m                    Model of substitution?    HKY (Hasegawa et al. 1985)  
t    Transition/transversion parameter?    Estimate from data set  
f                    Nucleotide frequencies?    Estimate from data set

RATE HETEROGENEITY

w                    Model of rate heterogeneity?    Uniform rate

Quit [q], confirm [y], or change [menu] settings:

Each option can be selected and/or edited by typing the corresponding letter. For example the user types b repeatedly, the options will change from tree reconstruction to Likelihood mapping to tree reconstruction again. The letter k cycles from quartet puzzling to user defined trees to pair-wise distances only (no tree). For example, a typical run to infer a tree based on DNA sequences would start with the following setting:

b                    Type of analysis?    Tree reconstruction  
k                    Tree search procedure?    Quartet puzzling  
v    Approximate quartet likelihood?    Yes  
u                    List unresolved quartets?    No  
n                    Number of puzzling steps?    10000  
j                    List puzzling step trees?    No  
o                    Display as outgroup?    L20571  
z    Compute clocklike branch lengths?    No  
e                    Parameter estimates?    Approximate (faster)  
x                    Parameter estimation uses?    Neighbor-joining tree

SUBSTITUTION PROCESS

d           Type of sequence input data?   Auto: Nucleotides  
h           Codon positions selected?   Use all positions  
m           Model of substitution?   HKY (Hasegawa et al. 1985)  
t    Transition/transversion parameter?   Estimate from data set  
f           Nucleotide frequencies?   Estimate from data set

RATE HETEROGENEITY

w           Model of rate heterogeneity?   Gamma distributed rates  
a    Gamma distribution parameter alpha?   Estimate from data set  
c           Number of Gamma rate categories?   8

Quit [q], confirm [y], or change [menu] settings:

By entering y the program computes a *quartet puzzling tree* based on 10000 intermediate trees, where the likelihoods for the *quartets* are only approximate likelihoods. The parameter estimates are also approximate estimates and are based on a *neighbor-joining* tree that TREE-PUZZLE computes automatically at the beginning of the optimization routine. The model of sequence evolution assumed is HKY (cf. section 4.6 and 4.9) and it is possible to estimate distances with  $\Gamma$ -distributed rate heterogeneity over sites, where the parameter is estimated with the aid of 8 discrete categories (cf. also section 4.6.1). If the settings above are confirmed, then the following output will appear on the screen:

Optimizing missing substitution process parameters  
Optimizing missing rate heterogeneity parameters  
Optimizing missing substitution process parameters  
Optimizing missing rate heterogeneity parameters  
Optimizing missing substitution process parameters

Optimizing missing rate heterogeneity parameters  
Optimizing missing substitution process parameters  
Optimizing missing rate heterogeneity parameters  
Writing parameters to file outfile  
Writing pairwise distances to file outdist  
Computing quartet maximum likelihood trees  
Computing quartet puzzling tree  
Computing maximum likelihood branch lengths (without clock)

All results written to disk:

Puzzle report file:	outfile
Likelihood distances:	outdist
Phylip tree file:	outtree

The computation took 299 seconds (= 5.0 minutes = 0.1 hours)

including input 1003 seconds (= 16.7 minutes = 0.3 hours)

The `outfile` summarizes the complete phylogenetic analyses and is very likely the most important file; in `outdist` is the matrix of pair-wise distances inferred from the model parameters; `outtree` describes the resulting *consensus tree* in the newick notation (see chapter 5 and figure 5.4). The content of the `outfile` is self-explanatory and it is not explained here (just give a look and see for yourself!).

A new feature recently implemented in TREE-PUZZLE version 5.0 allows to output all different tree topologies that have been found during puzzling steps. In the HIV example, 685 different intermediate trees have been found, where the most frequent tree occurred about 6.6%.

Therefore, the *quartet puzzling* algorithm can be used as a tool to generate a collection of plausible candidate trees, and this collection can subsequently be employed to search for the most likely tree among them. Note that the *consensus tree* does not necessarily coincide with the *maximum likelihood* tree and this is especially the case when the *consensus* is not fully resolved. If one is really interested in the *maximum likelihood* tree one should change option j to unique topologies, which will then be output in the file outptorder. A typical line of this file looks like

```
[ 1. 657 6.57 14 68510000](L20571,((AF10138,X52154),(U09127,(((U27426,U27445),(U067158,U09126)),
      ((U27399,U43386),(((L02317,AF042106),AF025763),U08443))))));
```

The first column is a simple numbering scheme, where each tree is numbered according to its frequency (second column, and third column). The column four gives the first time among 10000 (column 6) puzzling steps, when the tree was found. Column five shows how many different trees were found. To compute the *maximum likelihood* tree among all intermediate trees, rename outptorder into intree and run TREE-PUZZLE again with the following setting:

#### GENERAL OPTIONS

```
b           Type of analysis?  Tree reconstruction
k           Tree search procedure?  User defined trees
z           Compute clocklike branch lengths?  No
e           Parameter estimates?  Approximate (faster)
x           Parameter estimation uses?  Neighbor-joining tree
```

#### SUBSTITUTION PROCESS

```
d           Type of sequence input data?  Auto: Nucleotides
h           Codon positions selected?  Use all positions
m           Model of substitution?  HKY (Hasegawa et al. 1985)
```

t Transition/transversion parameter? Estimate from data set  
f Nucleotide frequencies? Estimate from data set

#### RATE HETEROGENEITY

w Model of rate heterogeneity? Gamma distributed rates  
a Gamma distribution parameter alpha? Estimate from data set  
c Number of Gamma rate categories? 8

Quit [q], confirm [y], or change [menu] settings: y

Now the program will compute the *maximum likelihood* values for all intermediate trees using the parameter estimates of the model of sequence evolution from the iterative procedure based on the *neighbor joining tree*. The computation will take some time, but it can provide more insights in the data and the reliability of the tree. The resulting `outfile` shows the likelihood computation for each tree topology and summarizes the results at the end of the file.

#### References

- ADACHI, J., AND M. HASEGAWA., 1995 MOLPHY: Programs for Molecular Phylogenetics, vers. 2.3, Tokyo, Institute of Statistical Mathematics.
- FELSENSTEIN, J., 1981 Evolutionary Trees from DNA Sequences: A *Maximum likelihood* Approach. *J Mol Evol* **17**: 368-376.
- FELSENSTEIN, J., 1993 PHYLIP: Phylogenetic Inference Package, Version 3.5c, Seattle Department of Genetics, University of Washington.
- FITCH, W. M., 1971 Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*:**20** 406-416.
- MARGUSH, T., 1981 Consensus n-trees. *Bulletin of Mathematical Biology* **43**: 239-244.

SAITOU, N., and M. NEI, 1987 The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees. *Mol. Biol. Evol.* **4**: 406-425.

STRIMMER, K., and A. VON HAESSELER, 1996a Quartet puzzling: A quartet *maximum likelihood* method for reconstructing tree topologies. *Mol Biol Evol* **13**: 964-969.

STRIMMER, K., and A. VON HAESSELER, 1996b Accuracy of neighbor joining for n-taxon trees. *Syst Biol* **45**: 514-521.

STRIMMER, K., N. GOLDMAN and A. VON HAESSELER, 1997 Bayesian probabilities and quartet puzzling. *Mol Biol Evol* **14**: 210-211.

SULLIVAN, J., K. E. HOLSINGER and C. SIMON, 1996 The Effect of Topology on Estimates of Among-Sites Rate Variation. *J Mol Evol* **42**: 308-312.

SWOFFORD, D.L., G.J. OLSEN, P.J. WADDELL, AND D.M. HILLIS, 1995 Phylogenetic inference, in: *Molecular Systematics* (eds. D.M. Hillis, C. Moritz, B.K. Mable), sinauer Associates Sunderland, 407-514.

UZZEL, T., and K.W. CORBIN, 1971 Fitting discrete probability distributions to evolutionary events *Science* **172**: 1089-1096.

WAKELEY, J., 1993 Substitution rate variation among sites in hypervariable region 1 of human mitochondrial DNA. *J. Mol. Evol.* **37**: 613-623.

YANG, Z., 1997 *Phylogenetic analysis by maximum-likelihood (PAML), version 1.3* Pennsylvania State University: Institute of Molecular Evolutionary Genetics.