Homework 2

Computational Biology course
Handed out: 17.10.2022
Due date: 31.10.2022 12:00 (midday)

Theory questions

We encourage you to answer these questions only after you have completed the programming task. Please keep your answers short, most of the questions can be answered in one or two sentences. Submit your answers in a pdf file named following the format Lastname_Firstname_HW2.pdf.

1. Imagine that the ancestral sequence of a tree of 10 species sampled at the present is composed of 50% T and 50% C. Assuming a Jukes-Cantor substitution model (all possible mutations occur at the same rate), what do you think the nucleotide frequencies at the leaves will be if the substitution rate is extremely **low** compared to the time scale of the tree? *(Only give a rough description—no calculation necessary!)*

2. For the same tree, what do you expect the nucleotide frequencies at the leaves to be if the substitution rate is extremely **high**?

3. Using your R code for simulating evolution, estimate ($\pm$100 mya) the minimal time t_{min} of evolution required under the TN93 model to reach a transition probability matrix P which no longer changes as time is increased. You can consider P to have stopped changing when its elements remain unchanged up to the 7^{th} decimal when you double the evolutionary time, i.e. $P(t_{min}) \simeq P(2t_{min})$. Use the values for α_1, α_2, β and π provided in the `test_simulation` function. The rates are already given in substitutions/mya.

4. Imagine you are trying to simulate evolution of a single site, occupied by nucleotide i, along a tree using only the substitution rate matrix Q, without ever computing the transition probability matrix P. The overall rate of change from a nucleotide i can be read as $-q_{ii}$ in the matrix Q. How could you randomly draw the time when the next substitution event happens?

5. Following question 4 and again using only the substitution rate matrix Q, assume you have drawn the time of the next substitution event for a particular nucleotide. How could you now sample the nucleotide it is substituted by?

Programming task

Submit your solution in an R file named following the format Lastname_Firstname.R.

Task description

In this homework you are going to simulate an alignment, i.e. a set of sequences with homologous sites, on a given tree. The objective is to simulate an alignment of *arbitrary* length under the TN93 model with an *arbitrary* mean substitution rate. The tree will be given as a string in Newick format. However, the handling of tree structure is already provided in the skeleton.

As input your code should take:

1. `N`: the number of sites in the alignment;
2. `pi`: the vector of equilibrium frequencies of nucleotides;
3. `alpha1`: relative rate of C $\leftrightarrow$ T transitions;
4. `alpha2`: relative rate of A $\leftrightarrow$ G transitions;
5. `beta`: relative rate of transversions.

As output the script should return an alignment with `N` sites of as many sequences as there are tips in the tree. A detailed step-by-step description and pseudocode overview of the algorithm are provided below.

Your script should follow the structure that we have laid out in the skeleton script called `CB_HW2_SequenceSimulation_skeleton.R`. This skeleton splits the assignment into smaller functions that are necessary to compute the final result. Each function is listed with the input and output that it requires, as well as a short description of its task. You just have to fill in the missing code (marked with `# ???`).

> Please do not change the structure of the code and the inputs and outputs of functions. The code will be tested and graded automatically and any structural changes will result in your code failing the tests.

To help you understand the skeleton structure, we provide a cross-reference table showing which functions in the skeleton should use which others (see Table 1).

In your assignment code, use numbers 1 to 4 to encode the nucleotides T, C, A, G (mind the order) during the simulation. The functions will be tested under the premise that the nucleotides are encoded this way. The final results should be in the form of a character string, however the necessary transformation function is included in the provided code.

We will also provide you with a test suite which is very similar to the one we will use to grade the homework. You can use that to verify that your code is performing as expected. Bear in mind that the suite used to grade your code will have a similar structure but will contain different parameters and may include more test conditions.

Function name	Uses

`simulate_evolution`	`create_TN93_Q_matrix`
	`get_starting_sequence`
	`get_evolved_sequence`
	`transform_to_nucleotides`

`get_starting_sequence`	`get_starting_nucleotide`

Table 1: Function cross-reference table

Simulation setup

Computing the substitution matrix

For TN93, the substitution rate matrix Q is defined as follows:

$$
\mathrm{Q}_{TN93} = (q_{ij})_{i,j\in\{\mathrm{T,C,A,G}\}} = \begin{array}{c c} & \begin{array}{cccc} T & C & A & G \end{array} \\ \begin{array}{c} T \\ C \\ A \\ G \end{array} & \begin{pmatrix} \cdot & \alpha_1\pi_C & \beta\pi_A & \beta\pi_G \\ \alpha_1\pi_T & \cdot & \beta\pi_A & \beta\pi_G \\ \beta\pi_T & \beta\pi_C & \cdot & \alpha_2\pi_G \\ \beta\pi_T & \beta\pi_C & \alpha_2\pi_A & \cdot \end{pmatrix} \end{array}
$$

This matrix specifies the rate of change from one nucleotide to another. The rows and columns of the matrix are ordered T, C, A, G, so that the rate of change from A→T is $\mathrm{q}_{AT} = \beta\pi_T$. The diagonals of the rate matrix, denoted with a dot, are set such that each row sums up to zero, e.g. $\mathrm{q}_{TT} = -(\alpha_1\pi_C + \beta\pi_A + \beta\pi_G)$.

The mean substitution rate (μ) of the matrix is:

$$
\mu = -\sum_{i\in\{T,C,A,G\}} \pi_i q_{ii}
$$

where π_i are the stationary frequencies of the nucleotides $i \in \{1, \ldots, 4\}$ and q_{ii} are the diagonal elements of the rate matrix Q.

```
Q = TN93(π, α1, α2, β);
for i = 1 to N do
    Sample a nucleotide n from distribution π;
    Append n to the root node sequence;
end
while not all branches have a sequence at the end do
    Get a branch b with a sequence S at the start and no sequence at
      the end;
    t_b = length(b);
    P(t_b) = e^(t_b Q);
    for each nucleotide n ∈ sequence S do
        Sample new nucleotide n_new from row n in P(t_b);
        Append n_new to the end sequence of branch b;
    end
end
```

Simulation of the starting sequence

For each site of the alignment the starting nucleotide is set using the equilibrium nucleotide frequencies. To get the starting nucleotides sample from the probability mass function that is defined by the equilibrium frequencies, e.g. $\pi = (\pi_T, \pi_C, \pi_A, \pi_G) = (0.22, 0.26, 0.33, 0.19)$.

Evolving the sequence

To evolve a site, we can use the transition probability matrices, as described in Lecture 3. For each branch b with branch length t_b we will need to compute the transition probability matrix $\mathrm{P}(t_b) = e^{t_b \mathrm{Q}}$. Thus, to see the result of evolution on nucleotide n along a branch of length t_b you need to sample from the probability mass function defined by row n in $\mathrm{P}(t_b)$, i.e. sample $\mathrm{P}(t_b)[n,]$.

This process is repeated for all branches in the tree. Importantly, always make sure that the evolution along a branch is initialised with the correct starting nucleotide – the nucleotide at the end of the previous branch.

Additional material and help

Required Packages

This homework assignment requires that the following pair of packages is installed on your R system:

ape : a library providing data types and methods suitable for phylogenetics, and

Matrix : a library containing various linear algebra functions.

To install these packages, simply enter the following command in the R command line:

```
install.packages(c("ape", "Matrix"))
```

Remember that you only need to install a package once and then you can load it for use in you code using the `library("PackageName")`. The homework skeleton loads the two aforementioned packages for you.

Useful functions

We provide a list of R functions that you may find useful when writing your code (Table 2).

Function name	Function description
`sample(values, size = 1, prob = probabilities)`	Get a single sample from a discrete probability distribution, where `values` is a vector of possible values and `probabilities` is a vector of appropriate probabilities, which together define the probability mass function.
`expm(matrix)`	Exponentiate the given `matrix`.
`rowSums(matrix)`	Return the vector of sums of the elements in each row of the given matrix.
`diag(matrix)`	Extract or replace the diagonal of a matrix, or construct a diagonal matrix.
`sapply(range, function)`	Apply a given function to a range of elements.

Table 2: Useful R functions

Additional information on tree representation

For testing purposes, you can use the tree of primates with branch lengths in units of million of years. The tree is shown in Figure 1, in Newick format the tree is specified as follows:
`(orangutan:13,(gorilla:10.25,(human:5.5,chimp:5.5):4.75):2.75);`

You can simulate an alignment down this primate tree by running the `test_simulation()` function defined in the skeleton.

In the function `simulate_evolution()`, we provide all the commands that you need for operating on the tree. A few of the functions that we use and the tree structure are implemented in the package `ape`, which is loaded at the top of the skeleton. The function `read.tree()` reads in the phylogenetic tree represented in Newick format, and the tree is then reordered by clades by the function `reorder()`. Each node in the tree, including internal nodes, gets assigned a numeric label for reference. The primate tree is represented as shown in Figure 2.

The `tree` object within `ape` fully describes the phylogenetic tree. `tree$edge` contains the branches of the tree as directional edges between nodes (from parent to child) in the form of a matrix, where each row is an edge and the first column represents

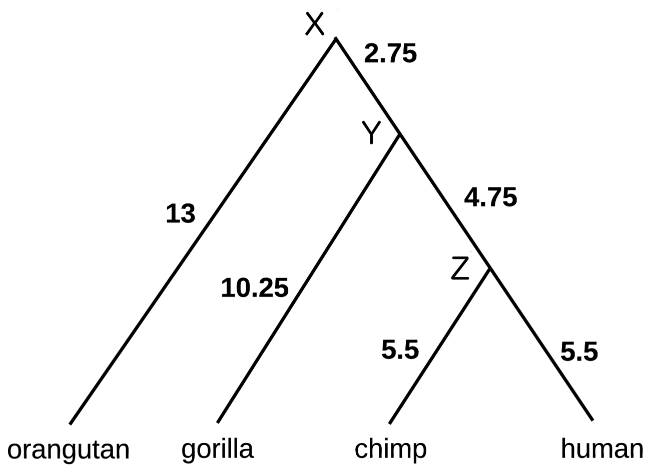


Figure 1: Primate tree

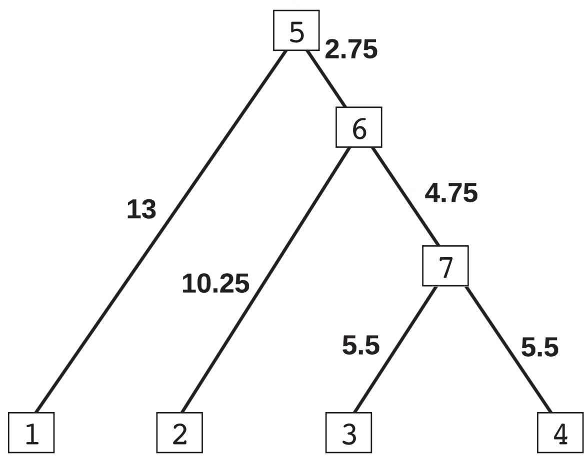
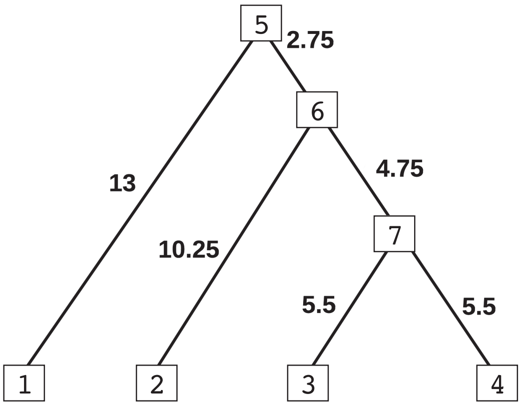


Figure 2: Primate tree with node labels attributed by `ape`.

parent nodes and the second represents child nodes. The branch lengths are stored in `tree$edge.length`, which is a vector of numeric values that is ordered according to the rows in `tree$edge`. Both of these structures are illustrated in Figure 3. Thus, for example, if you would like to access the length of the branch from node `6` to node `7`, which is stored in the fourth row of the edge matrix: `tree$edge[4, ]`, you would need to read the fourth element in the edge length vector: `tree$edge.length[4]`, which would yield the value `4.75`.

To make your life easier we also provide a list `sequence_per_node` which stores the nucleotide sequences per node. It is designed to use the node labels defined by `ape` and store the sequences in the appropriate way.



tree$edge		tree$edge.length
parent	child	
5	1	13
5	6	2.75
6	2	10.25
6	7	4.75
7	3	5.5
7	4	5.5

Figure 3: Primate tree with the associated branch description in `ape`.

Additional reading

1. Computational Molecular Evolution by Ziheng Yang, chapter 9, section 9.5, in particular section 9.5.1.3 (p.302-304)

2. `http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780198567028.001.0001/acprof-9780198567028-chapter-9`

3. Bayesian Analysis of Molecular Evolution using MrBayes by John P. Huelsenbeck and Fredrik Ronquist, chapter 3, section 3.1 (p.5,7-9) `http://www.molecularevolution.org/molevolfiles/mrbayes/mrbayes_2004.pdf`