MASTERARBEIT / MASTER’S THESIS

Titel der Masterarbeit / Title of the Master‘s Thesis

„EvoNAPS: A Database for Natural Parameter Settings of Evolutionary Models“

verfasst von / submitted by

Franziska Reden BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 875
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Bioinformatik
Betreut von / Supervisor:	Univ.-Prof. Dr. Arndt von Haeseler
Mitbetreut von / Co-Supervisor:	Dr. Heiko Andreas Schmidt

Abstract

Sequence simulations play a vital role in phylogenetic research as they enable the evaluation of phylogenetic models or methods. Moreover, with the increasing impact of machine learning algorithms the simulations provide the huge amount of data required for their training. To ensure that the simulated sequences are as realistic as possible the simulations should be based on empirical data.

For this purpose, EvoNAPS, a database providing NAtural Parameter Settings of EVOultionary models, was designed and implemented. Over 29,000 biological alignments from three different published sources were gathered. The alignments were then analysed using the phylogenetic inference software IQ-Tree 2. From the results a huge number of features regarding the alignments, the inferred phylogenetic trees as well as the respective parameter estimates of the evolutionary model were extracted and stored in the EvoNAPS database.

To date EvoNAPS holds over 64,000 phylogenetic trees and the respective model parameter estimates. EvoNAPS allows the retrieval of typical parameter settings for 286 different DNA and 364 different protein models. The database comes equipped with various filter options that allow the user to find model parameter estimates, alignments and/or trees that closely match the data they want to simulate.

Overall, the EvoNAPS database represents a valuable resource to those studying model-based phylogenetics and will greatly aid and facilitate future phylogenetic studies.

Zusammenfassung

Das Simulieren von Sequenzen spielt eine entscheidende Rolle in der phylogenetischen Forschung, da es das Evaluieren von phylogenetischen Methoden oder Modellen ermöglicht. Außerdem kann die große Menge an Daten, die für das Trainieren von sogenannten „Machine-Learning-Algorithmen“, die immer mehr an Bedeutung gewinnen, durch Sequenzsimulationen generiert werden. Um sicherzugehen, dass die simulierten Sequenzen so realistisch wie möglich sind, sollten die Simulationen auf empirischen Daten basieren.

Aus diesem Grund wurde EvoNAPS, eine Datenbank für Parameter-Einstellungen von evolutionären Modellen und für phylogenetische Bäume, die auf empirischen Daten basieren, entworfen und implementiert. Insgesamt wurden über 29.000 biologische Alignments aus drei unterschiedlichen veröffentlichen Quellen gesammelt. Die Alignments wurden mithilfe des IQ-Tree2 Programms, einer Software für phylogenetische Inferenz, analysiert. Eine große Menge an Eckdaten und Merkmalen der analysierten Alignments sowie der abgeleiteten Bäume und dazugehörigen Modellparametern wurde gesammelt und in der EvoNAPS Datenbank gespeichert.

EvoNAPS enthält über 64.000 phylogenetische Bäume und die dazugehörigen geschätzten Modellparameter. Die Datenbank bietet typischen Parameter-Einstellungen von 286 unterschiedlichen DNA und 364 unterschiedlichen Protein Modellen. Außerdem ist die Datenbank mit verschiedenen Filteroptionen ausgestattet, die es den Nutzenden erlauben, Alignments, Bäume und/oder Parameter-Einstellungen von evolutionären Modellen zu finden, die den zu simulierenden Daten möglichst ähneln.

Die EvoNAPS Datenbank stellt eine nützliche Ressource für all jene dar, die an modellbasierter Phylogenie forschen, und wird eine große Hilfestellung in zukünftigen phylogenetischen Studien sein.

Contents

Chapter 1

Introduction and Motivation

Phylogenetics is the study of evolutionary relationships between biological entities such as species or individual organisms (Felsenstein, 2004). While in the past such relationships were studied based on morphological data, nowadays mainly sequence data is being used. The key assumptions in phylogenetics are that the sequences studied are homologous, meaning they share a common ancestor sequence, and that the observed differences between sequences are due to random mutation events that occurred during evolution. The sequences are arranged into an alignment with n sites, whereas each aligned site can be regarded as a sample of evolution (e.g., Mount, 2004, pp.183-226, 284-327). The variation within the sampled sites reflect the evolutionary distance between the sequences.

The results of phylogenetic analysis is usually a *phylogenetic tree* that was reconstructed based on the alignment (e.g., Felsenstein, 2004; Graur and Li, 2000). The leaves in the tree represent the taxa (or sequences) that are connected with each other via branches, whereas the length of the branches reflect the evolutionary distances between sequences.

The inner nodes in a tree depict specification events (also called splits) and represent a common ancestors sequence.

The number of possible trees that vary in their tree topologies (i.e., their conformation) drastically increases with the number of sequences in the alignment (Felsenstein, 1978). The challenge is to find the tree that best explains the data. This prompts a probabilistic view on the trees. To calculate how probable it is to observe the data given a specific tree requires an explicit *model of sequence evolution* as first described by Jukes and Cantor (1969).

Typically sequence evolution is modelled according to a continuous-time Markov process, whereas the mutation events are assumed to be Poisson-distributed (Bryant et al., 2005). The states of the Markov process are either the $c = 4$ nucleotide bases for DNA alignments or $c = 20$ amino acids for protein alignments. Each state x occurs with a assumed frequency π_x with $\sum_{x=1}^{c} \pi_x = 1$. Changes (or substitutions) from any state x to any other state y are modelled to occur with a relative substitution rate ρ_{xy}. Furthermore, these models make some fundamental assumptions regarding sequence evolution, namely that sites evolve *independently* and that the Markov process is *reversible* and *stationary*.

Various of these substitution models have been described in literature. (A list of models can be viewed in the appendix in tables A.1 and A.2.) Furthermore, these models can be extended to additionally incorporate rate heterogeneity among sites (RHAS). Popular RHAS models are invariable sites models (*+I*; Churchill et al., 1992), Gamma models (*+G*; Yang, 1994), a combination of the two (*+I+G*; Gu et al., 1995) or *free-rate* models assuming k rate categories (*+Rk*; Soubrier et al., 2012).

Among the plethora of models that have been described any can be chosen to be used in phylogenetic tree reconstruction. The branch lengths of a tree inferred with an explicit model of evolution correspond to the *expected number of mutations per site*. Methods that use an explicit model for the inference of a tree are also referred to as model-based methods. Among those, *maximum likelihood* (ML) methods as described for phylogenetics by Felsenstein (1981) are very popular (Bryant et al., 2005). The key idea behind ML methods is to assign likelihoods to each tree and to find the tree that has maximum likelihood (ML tree).

Figure 1.1a depicts a typical model-based phylogenetic inference workflow. The first step is to choose a fitting model of sequence evolution based on the alignment. This step is also referred to as the *model selection* step. The second step entails the tree search based on the alignment and chosen model.

A great challenge in the elaborated phylogenetics is that usually the true evolutionary process underlying the studied sequences is unknown (Felsenstein, 2004). Accordingly, the validation of the methods and models that have been used as well as the results they produced is virtually impossible. Hence, phylogenetic studies have come to heavily rely on *simulated sequences* as for them the underlying truth *is* known (e.g., Garland et al., 1993; Tateno et al., 1994; Gaut and Lewis, 1995; Hillis, 1995; Kupczok et al., 2010). Furthermore, with an increasing interest in machine learning algorithms, such as Neural Networks (NN), sequence simulations can be used to provide the huge amount of training data required.

Due to the widespread use and necessity of simulated alignments, it is of great importance to choose the input for the simulations carefully. Typically, sequence simulations require a phylogenetic tree and values for the parameters of the chosen model of sequence evolution as input (see figure 1.1b). To make sure that the simulated sequences

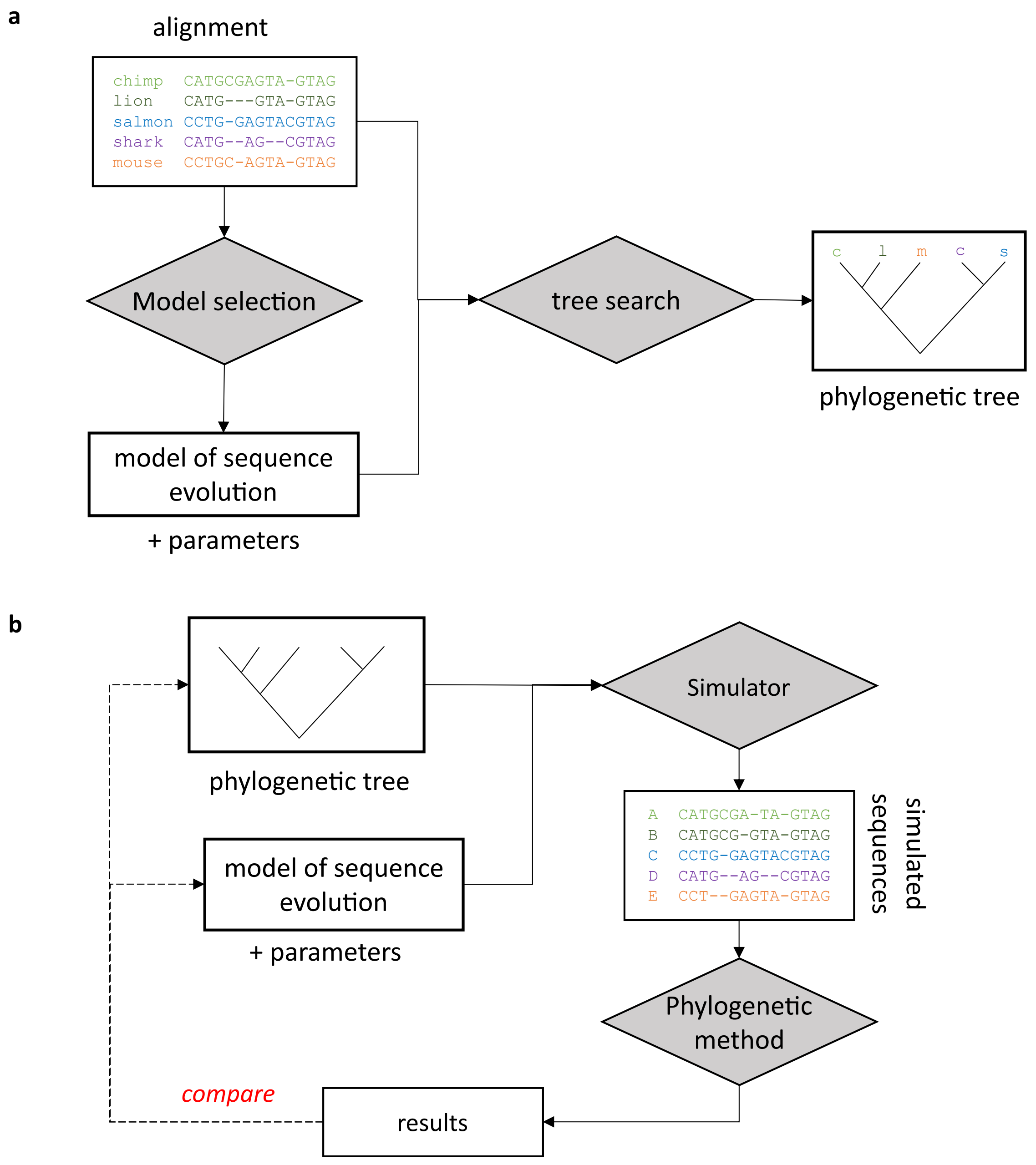

Figure 1.1: Workflows of phylogentic analysis: Figure **(a)** schematically depicts a typical model-based phylogenetic inference workflow. In the first step a model of sequence evolution is selected. The second step involves the tree search based on the alignments and the chosen model. Figure **(b)** sketches a typical workflow for generating simulated sequences. The simulation of sequences typically requires a phylogenetic tree and an explicit model of sequence evolution with its parameters as input. The generated sequences can in turn be used to evaluate phylogenetic methods.

are not completely uncoupled from biological data, the input tree and model parameters should be based on empirical data. However, it is unclear how parameter estimates of the different evolutionary models and the branch lengths of the inferred trees are distributed in biological data.

Here, we present EvoNAPS, a database for NAtural PArameter Settings of EVOlutionary models. The database provides parameter estimates based on biological data of 286 different DNA and 364 different protein models as well as over 32,000 ML trees. The data for the database was created by applying a typical model-based phylogenetic inference workflow (see figure 1.1a) on biological alignments that were gathered from published sources.

Various features of each evaluated alignment and the inferred tree were gathered. Example features are the number of sequences, sites or patterns for the alignment or tree length, tree diameter and mean branch length for the tree. (A full list of gathered features is provided in section 2.2.1 of the next chapter.) The gathered features can be used to filter for alignments, model parameter estimates and/or trees that are of interest to the user and are (based on) biological data.

The remainder of this chapter will give some additional insight on models of sequence evolution and how they are parameterized (section 1.1). Furthermore, model selection will be discussed in more detail in section 1.2.

In chapter 2 the sources of the gathered alignments (section 2.1) as well as the working steps required to create the data for the EvoNAPS database will be discussed in detail (sections 2.2). The architecture of the database will be elaborated on in section 2.3.

Example feature extractions from the EvoNAPS database will be given in chapter 3.

1.1 Models of sequence evolution

This section will give more detailed information on models of sequence evolution. Note, that it only focuses on the information that is relevant for this master thesis project. For a more in-depth education refer to, e.g., chapter 5 in "Fundamentals of Molecular Evolution" by (Graur and Li, 2000, pp.165-248) or chapter 5 in the book "Molecular evolution" by (Li, 1997, pp.99-148). This section here will mainly follow the nomenclature and narrative as found in the chapter "Likelihood calculations in molecular Phylogenetics" by Bryant et al. (2005) of the book "Mathematics of Evolution and Phylogeny" by Gascuel (2005).

As has already been discussed above, there exists a plethora of evolutionary models that have been described in literature (see tables A.1 and A.2 in the Appendix). While there are simpler, tractable models as well as more complex and parameter-rich models, it needs to be pointed out that all models are naturally approximations and simplifications of the actual evolutionary process. In reality, evolution is so complex that a completely accurate calculation of probabilities is impossible. However, even with the simplified models as approximations remarkable results can be achieved (Bryant et al., 2005).

The most widely used models of evolution, and therefore the models of interest in this thesis, are part of a restrictive class of Markov models. These models make a certain set of assumptions regarding sequence evolution that will be discussed in 1.1.1 and can be described with a set of parameters as discussed in 1.1.2 and 1.1.3.

1.1.1 Assumptions of models of sequence evolution

The evolutionary models covered in this master thesis project model mutations events according to a continuous-time Markov chain with the number of mutations being Poisson distributed. These Markov models are *restrictive* in the sense that they make certain assumptions regarding sequence evolution, namely that sites evolve independently, and that the Markov process is stationary and time reversible (Bryant et al., 2005).

Let us assume an alignment with n sites. Each site x can be in any of the defined states with the number of states being $c = 4$ for DNA alignments or $c = 20$ for protein alignments. Each state x occurs with a assumed frequency of π_x with $\sum_{x=1}^{c} \pi_x = 1$. Substitutions from any state x to any other state y are modelled to occur with a relative substitution rate of ρ_{xy}.

Independence of sites. The first simplifying assumption states that all sites n along a sequence evolve independently. The probability of one sequence A evolving into sequence B, therefore, equals the product of site probabilities across all sites n.

Stationarity. As time goes to infinity, the state frequencies will reach an equilibrium, or stationary distribution $\boldsymbol{\pi}$. If the initial frequencies of the sequence at the root of a phylogenetic tree are in the stationary distribution, there will be no (observable) change in state frequencies across the whole tree. The restrictive models covered here all assume a stationary process and, therefore, assume that all node distributions in a tree equal the stationary distribution, including the root and the leaves.

Time reversibility. The third assumption is that the substitutions in a sequence are reversible. This means that the substitution rate from state x to state y equals the substitution rate from state y to state x, $\rho_{xy} = \rho_{yx}$. Time reversibility greatly simplifies the computation of likelihoods on a tree as the likelihood becomes independent of the position of the root.

1.1.2 Parameterization of the substitution model

Let us assume we want to model the evolution of a set of DNA sequences using a model that fulfills the assumptions as discussed in 1.1.1. The possible states in the Markov model correspond to the four nucleotide bases A, C, G and T. We assume stationarity across the whole tree with equilibrium state frequencies $\boldsymbol{\pi} = (\pi_A, \pi_C, \pi_G, \pi_T)$ and allow substitutions from any state x to any state y with $x, y \in (A, C, G, T)$. As we assume time-reversibility this accounts for the six different substitution rates $\rho_{AG}, \rho_{AT}, \rho_{CG}, \rho_{CT}, \rho_{GT}$. By convention the substitution rates are set to be relative to rate $\rho_{GT} = 1$. Note, that substitutions from state x to the same state x are being modelled, but to assure a stationary distribution across the tree their rates cannot be chosen arbitrarily. This might be better understood when looking at formula 1.1 below. The so-called instantaneous rate matrix Q combines the equilibrium state frequencies $\boldsymbol{\pi}$ and substitution rates into a single rate matrix:

$$Q = \begin{pmatrix} * & \rho_{AC}\pi_C & \rho_{AG}\pi_G & \rho_{AT}\pi_T \\ \rho_{AC}\pi_A & * & \rho_{CG}\pi_G & \rho_{CT}\pi_T \\ \rho_{AG}\pi_A & \rho_{CG}\pi_C & * & \rho_{GT}\pi_T \\ \rho_{AT}\pi_A & \rho_{CT}\pi_C & \rho_{GT}\pi_G & * \end{pmatrix} \tag{1.1}$$

Each row in Q stands for the initial state x, whereas the columns indicate the final state y for any substitution from x to y. Note, that the diagonal entries of the matrix, marked with $*$, must be chosen such that the rows sum up to zero. This ensures, as already mentioned above, that the assumption of a stationary distribution across the tree is respected. The diagonal entries do not hold biological meaning but are a mathematical convenience.

Overall, there are ten parameters in the rate matrix Q, namely the six substitution rates and the four equilibrium frequencies. Because by convention the rates are normalized to $\rho_{GT} = 1$ and the equilibrium frequencies need to sum up to 1, the number of free parameters is reduced to eight. Different evolutionary models vary in the number and kinds of restrictions they pose on the rate matrix Q.

Take for example the simplest model, the Jukes Cantor (JC) model (Jukes and Cantor, 1969). The JC model assumes equal state frequencies for the stationary distribution, $\pi_A = \pi_C = \pi_G = \pi_T = 0.25$, and equal substitution rates. It is the most restrictive model by restricting all frequencies and rates to be equal and, therefore, has zero degrees of freedom.

A slightly more complex and less restrictive model is the F81 model, named after its creator and creation year (Felsenstein, 1981). The F81 model allows for unequal state frequencies, $\pi_A \neq \pi_C \neq \pi_G \neq \pi_T$, but keeps equal substitution rates. Hence, the F81 model has three degrees of freedom.

The two models mentioned so far assume that each type of substitution occurs with the same rate. However, in reality mutations between the same type of base (transitions) happen with higher frequency than between different types of bases (transversions). For example, mutations from the *purine* base adenine (A) to the *purine* base guanine (G) happens with higher frequency than from *purine* base adenine to the *pyrimidine* base cytosine (C). Taking these biological observations into account, the Kimura two parameters (K2P) model (Kimura, 1980) differentiates between transition and transversion rates, $\rho_{AG} = \rho_{CT}, \rho_{AC} = \rho_{AT}$ vs. $\rho_{CT} = \rho_{GT}$. The K2P model assumes equal state frequencies and rate ρ_{GT} is set to 1. Thus, the model has one free parameter, the transition rate.

The most complex and least restrictive model that respects the assumptions mentioned in 1.1.1 is the General Time Reversible (GTR) model (Tavaré, 1986). The GTR model assumes unequal state frequencies and allows for six different relative substitution rates. Therefore, the model has 8 degrees of freedom.

Many more substitution rate matrices Q, besides the four already mentioned, have been described in literature. In this master thesis project we focus on the 22 named substitution rate matrices that are frequently used and implemented in most phylogenetic inference software (see A.1).

Even though only substitution rate matrices of DNA sequence evolution models were discussed so far, protein models can be described with similar rate matrices. When modelling evolution of protein sequences, Q is a 20×20 matrix corresponding to 20 amino acids as states. Accordingly, there are 190 possible substitutions given the assumed time-reversibility.

In practice substitution rates and state frequencies (given that they are modelled to be unequal) in DNA models are estimated based on the alignment given the manageable size of free parameters they produce. For protein models, however, the number of parameters to be estimated is so large that in practice rate matrices with fixed substitution rates and state frequencies are used. Sometimes the state frequencies are estimated from the input data. Frequently used protein substitution rate matrices are listed in table A.2.

1.1.3 Rate heterogeneity among sites (RHAS)

The evolutionary models mentioned so far assume that the substitutions across all sites in a sequence occur with the same rate. However, this assumption is highly unrealistic. In biological data sets there are typically fast and slow evolving sites corresponding to functional, highly conserved regions and less conserved regions in a sequence respectively (Bryant et al., 2005).

One approach to incorporate substitution rate heterogeneity among sites (RHAS) is to use invariable sites models (Churchill et al., 1992). The model extension (*+I*) assumes that a certain proportion of sites in an alignment is invariable (representing a subset of constant sites). The remaining sites are assumed to evolve at the same rate. Invariable sites models introduce an additional model parameter, which is the assumed proportion of invariable sites.

Another popular approach is to model RHAS according to a Gamma (Γ) distribution (Yang, 1994). In practice, the rates that are assumed to follow a Γ-distribution are approximated using a discrete number of rates k (typically, between 4 and 8 rates). The degree of influence of each rate category in comparison to each other is set to be equal by cutting the Γ-distribution into k parts of equal size (equal in regards to the area under the curve). The mean (or alternatively median) rates of each rate category are then used to approximate Γ-distributed rates among sites. Gamma models (*+G*) can be described with a single additional parameter, the shape parameter α of the assumed Γ-distribution.

Should the number of invariable sites be non-negligible, a combination of the invariable sites model and Gamma model (*+I+G*) has been shown to be a good approach for estimating the variation of substitution rate among sites (Gu et al., 1995).

Even though Gamma models tend to be a well performing approximation, biological data does not necessarily follow Γ-distributed substitution rates (Huelsenbeck and Hillis, 1993). Alternatively, RHAS can be modelled using, again, a number of discrete rates k, but they do not have to follow a Γ-distribution or in fact any specific distribution. The proportion (or degree of influence) of each rate category can also be chosen freely and are not assumed to be equal like in the Gamma model. These so-called probability-distribution-free or *free-rate* (*+R*) models (Soubrier et al., 2012) introduce $2k$ additional parameters to the evolutionary model, corresponding to the proportion and the rate of each rate category.

Any given substitution rate matrix (see tables A.1, A.2) can be paired with any model of RHAS as discussed above (*+I,+G,+I+G,+R*), resulting in a huge number of different evolutionary models. Selecting the optimal model is a critical step in phylogenetic analysis that can greatly affect the maximum likelihood estimation (Johnson and Omland, 2004). Given the large set of model candidates, model specification is a non-trivial matter (Kalyaanamoorthy et al., 2017; Posada, 2008), which will be discussed in the next section.

1.2 Model selection

The first step in a model-based phylogenetic analysis is to choose a model of sequence evolution. To do so, the candidate models need to be evaluated in a process termed model selection or model evaluation. The aim of model selection is to find the best model among a set of model candidates (Kalyaanamoorthy et al., 2017; Posada, 2008).

Model selection is a problem not restricted to phylogenetics. Generally, a model performs well if it manages to describe the observed data with high accuracy and, therefore, can be regarded as a good approximation of the true underlying process. In that sense, complex models will perform well due to their flexibility. However, there is a trade-off between *descriptive accuracy* and *parsimony* (Burnham and Anderson, 2001; Wagenmakers and Farrell, 2004). Parsimony hereby stands for being conservative in regards of the number of parameters that need to be estimated. This trade-off exists as with each additional parameter the vulnerability of the model to random error increases. Accordingly, it is not desirable to always choose the most complex model. The best model should be the one that describes the data with adequate accuracy while using a minimum number of parameters.

1.2.1 Selection criteria

In phylogenetic analysis, a still popular method for comparing multiple models is the Likelihood Ratio Test (LTR; Solomon, 1975). However, LTR bears the disadvantage that the models that are being compared need to be nested in each other. Furthermore, LTR tends towards choosing the most complex model (Posada and Buckley, 2004).

Alternative and widely used methods for model selection that are not restricted to nested models are information criteria. Akaike (1973) has shown that choosing the model with lowest expected information loss is asymptotically equivalent to choosing a model that has the lowest Akaike information criterion (AIC) value. The AIC value of a model is easily computed given the likelihood L and the number of free parameters k. AIC is defined as

$$AIC = -2 \ln L + 2k. \tag{1.2}$$

Equation 1.2 shows how AIC rewards descriptive accuracy by incorporating L, and penalizes lack of parsimony according to the number of free parameters k (in blue). Here, the number of free parameters is given by the number of branch lengths and the number of model parameters that need to be estimated. In a fully resolved tree (containing only bifurcations) with s leaves (or taxa) the number of branches is given by $k_B = (2 * s - 3)$.

It has been shown that AIC is a good approximation for sufficiently large data sets. However, for smaller data sets, with $n/k < 40$, a finite sample correction is recommended. The corrected AIC (AIC$_c$; Burnham and Anderson, 2003) is defined in equation 1.3. Here, the sample size n corresponds to the number of sites in the alignment.

$$AIC_c = -2\ln L + 2k + \frac{2k * (k+1)}{n-k-1} \tag{1.3}$$

Despite its popularity and wide-spread use, some believe that AIC is too liberal and tends to select overly complex models while neglecting sampling variability (Kass and Raftery, 1995). Furthermore, AIC is not consistent in the sense that with a large number of observations n, AIC tends to fail to recover the true low-dimensional model (Bozdogan, 1987).

A popular alternative to the AIC is the Bayesian information criterion (BIC; Schwarz, 1978). BIC is defined as:

$$BIC = -2\ln L + k * \ln n \tag{1.4}$$

BIC is an asymptotic approximation to the Bayesian model selection and is consistent as $n \to \infty$ (Schwarz, 1978).

Although the selection criteria mentioned so far are similar algebraically, they are motivated by different theories. There is no definite answer to the question which selection criteria is the best. The choice depends on the goal the researcher wants to achieve, higher accuracy or parsimony. Generally, if the emphasis lies on achieving a good prediction, AIC or AIC_c tend to be the better choice for model selection but might choose overly complex models (overfitting). On the other hand, if the emphasis lies on parsimony, BIC might the better choice but one has to accept the likely error of underfitting (Dziak et al., 2020).

1.2.2 Akaike weights

All selection criteria mentioned so far can be used to select the best model among a model candidate set. However, the value of the selection criterion alone does not give any insights regarding to what extent the data supports the best model in comparison to the remaining evaluated models. For this purpose, the so-called Akaike weights can be calculated (e.g., Burnham and Anderson, 2001; Wagenmakers and Farrell, 2004).

To calculate the Akaike weight of model i, first the difference between the AIC of the best model (the model with minimum AIC) and the AIC value of model i needs to be calculated:

$$\triangle_i(AIC) = AIC_i - minAIC \tag{1.5}$$

The Akaike weights are then calculated according to the following formula:

$$w_i(AIC) = \frac{\exp\{-\frac{1}{2}\triangle_i(AIC)\}}{\sum_{k=1}^{K}\{-\frac{1}{2}\triangle_k(AIC)\}} \tag{1.6}$$

Akaike weights can be interpreted as the probability that model i is the best model given the data and the set of candidate models (Wagenmakers and Farrell, 2004). Furthermore, the strength of evidence in favour of one model i over another model j can be calculated by dividing their Akaike weights, $\frac{w_i}{w_j}$.

Weights for AIC$_c$ can also be calculated by replacing the AIC values in equations 1.5 and 1.6 with AIC$_c$ values. The same applies to BIC values, whereas BIC weights are also referred to as “Schwarz” weights.

1.2.3 The *ModelFinder* algorithm

In this project we decided to use the IQ-Tree2 software as it offers the greatest number of different evolutionary models (including *free-rate* models) and has integrated the

ModelFinder algorithm (Kalyaanamoorthy et al., 2017) for efficient model selection. Here, the *ModelFinder* algorithm will be discussed in detail.

There exists a vast number of evolutionary models that can be used for tree reconstruction as has been discussed in section 1.1. Ideally, a tree search is conducted using each available evolutionary model to find the ML tree. However, this would be computationally exhaustive and would take an often unreasonable amount of time. Therefore, heuristics are employed to find a model M among the model candidate set that fits the alignment sufficiently well. The chosen model M will then be used in the subsequent ML tree search. The *ModelFinder* algorithm (Kalyaanamoorthy et al., 2017) is such a heuristic.

The idea behind the *ModelFinder* algorithm is to first build a fast, but reasonable tree T based on the data D (further also referred to as *initial tree*). By default, tree T is built based on the model *GTR+I+G4* for DNA alignments and *LG+I+G4* for protein alignments. The likelihood L of each model M in the model candidate set is evaluated on T; $L = (T, M|D)$. For the likelihood calculations the branch lengths of tree T are allowed to be re-estimated but the tree topology stays the same to make the likelihoods calculations of each model comparable. The best-fit model is then decided according to their BIC values by default. However, the selection criterion can also be changed to either AIC or AIC$_c$.

The candidate models available in *ModelFinder* also include *free-rate* models. Should *free-rate* models be evaluated the default algorithm as discussed above is slightly modified. In this modified version there is a maximum number of *free-rate* categories k_{max} that is being considered. After building a reasonable tree T, the likelihoods $L = (D|T, M_k)$ of the *free-rate* models with k rates (*+Rk*) are consecutively calculated until k_{max} is reached, starting with $k = 2$. However, should the model with less *free-*

rates perform better than the model with one more *free-rate*, $BIC(M_k) > BIC(M_{k-1})$, then the algorithm stops evaluating any more *free-rate* models. This stopping condition is based on the assumption that should model M_{k+1} perform worse than model M_k, then modelM_{k+2} will also fail to perform better.

Chapter 2

Resources and Methods

Building the EvoNAPS database entailed three separate working steps. First, biological alignments from published sources were gathered (see section 2.1). Secondly, a scientific workflow (Taylor et al., 2007) was created that applied to each alignment selects the best-fit model from a vast model candidate set and subsequently builds a phylogenetic tree (see section 2.2). The third part included designing and implementing the EvoNAPS database (see 2.3).

2.1 Gathering Alignments

In the course of this project alignments from three different sources were analysed. One of them is the online repository “BenchmarkAlignments” consisting of 67 multi-gene data sets provided by Rob Lanfear available on GitHub (Lanfear, 2019). Of those we used 31 DNA data sets that were cut into declared partitions using the Python script (`AMAS.py`) provided by Rob Lanfear. The resulting 1,839 partitions were scanned for sequences only containing gaps which were excluded from the alignments.

The second source is the OrthoMaM database (v. 10c Douzery et al., 2014), a databases for mammalian marker sequences. Overall, 14,345 DNA alignments from OrthoMaM were gathered.

The third source for alignments is the PANDIT (v. 17.0) database (Whelan et al., 2006), a collection of multiple sequence alignments and phylogenetic trees that cover common protein domains and their coding DNA sequences. In total 6,491 DNA alignments and 6,614 protein alignments from the PANDIT database were analysed. Note that alignments containing 3 or less sequences were discarded as they were deemed to not contain enough information for proper parameter estimation.

2.2 Data Generation

A typical model-based pyhlogenetic workflow was applied on each of the alignments gathered. The workflow includes a model selection step followed by a ML tree search based on the best-fit model. The workflow will be discussed in detail in section 2.2.

2.2.1 Overview of the gathered features

From the alignments as well as the output produced by the workflow a multitude of features are selected to be imported into the EvoNAPS database. The features can be classified according to the entity they describe, namely the alignment, the sequences in the alignment, the evaluated models during model selection, the (ML or *initial*) trees, and the branches of the trees. The tables below list the selected features accordingly.

Table 2.1 shows the extracted features of the gathered alignments. Table 2.2 shows the features of each sequence in the gathered alignments.

The gathered features of each model that was evaluated during model selection are described in table 2.3. Note, that some features are model specific (e.g, a proportion

Table 2.1: Alignment features stored in the EvoNAPS database.

	feature	**details**
1	# sequences	
2	# sites (columns)	
3	# parsimony informative sites	
4	# singleton sites	
5	# constant sites	
6	fraction of wildcards and gaps	
7	# distinct patterns	
8	# sequences that failed the Chi2 test	Chi2 test to evaluate whether nucleotide composition of the sequence matches nucleotide composition of the alignment.
9	# identical sequences	
10	# excluded sequences	as will be further elaborated on in section 2.2.2.1

Table 2.2: Sequence features stored in the EvoNAPS database.

	feature	**details**
1	Sequence name	as it appear in the original alignment
2	fraction of wildcards and gaps	
3	whether the sequence passed or failed the chi2 test	Chi2 test to evaluated whether nucleotide composition of the sequence matches nucleotide composition of the alignment.
4	whether the sequence was excluded from calculations	as will be further elaborated on in section 2.2.2.1
5	name of the sequences(s) the sequence is identical to	
6	empirical state frequencies	bases (DNA) or amino acids (protein)
7	sequence	as it appears in the original alignment (with wildcards and gaps)

of invariable sites, alpha value for Gamma models). The features also include the calculated weights (of to the different information criteria as described in section 1.2) of the model. The tree length (feature 19) was calculated based on the evaluated model and is that of the *initial* tree. Note, that the tree topology of the *initial* tree remains

Table 2.3: Features of the evaluated models stored in the EvoNAPS database.

	feature	**details**
1	model	evaluated model
2	substitution rate matrix	
3	RHAS model	
4	# rate categories	
5	logL	the logarithmic likelihood of the *initial* tree based on the evaluated model
6	AIC value	Akaike (1973)
7	AIC weight	
8	AICc value	Burnham and Anderson (2003)
9	AICc weight	
10	BIC value	Schwarz (1978)
11	BIC weight	
12	CAIC value	consistent Akaike information criterion (Bozdogan, 1987)
13	CAIC weight	
14	ABIC value	adjusted Bayesian information criterion (Sclove, 1987)
15	ABIC weight	
16	# free parameters	
17	# free model parameters	
18	# branches	
19	tree length	of the *initial* tree
20	proportion of invariable sites	
21	shape parameter alpha	
22	state frequencies of the assumed stationary distribution	states are bases (DNA) or amino acids (protein)
23	assumed relative substitution rates	for DNA 6 rates, for protein 190 rates
24	proportion of the assumed rate category	of up to ten rate categories (*+R10*)
25	rate of the assumed rate category	of up to ten rate categories (*+R10*)

unchanged during the evaluation of the models, but that the branch lengths are re-estimated based on the evaluated model. Hence, the tree length can vary between evaluated models.

Table 2.4: Tree features stored in the EvoNAPS database.

	feature	**details**
1	model	model used in ML search or for the *initial* tree
2	substitution rate matrix	
3	RHAS model	
4	# rate categories	
5	logL	the logarithmic likelihood of the tree
6	unconstrained logL	
7	AIC value	Akaike (1973)
8	AIC_c value	Burnham and Anderson (2003)
9	BIC value	Schwarz (1978)
10	CAIC value	consistent Akaike information criterion (Bozdogan, 1987)
11	ABIC value	adjusted Bayesian information criterion (Sclove, 1987)
12	# free parameters	
13	# free model parameters	
14	# branches	
15	proportion of invariable sites	
16	shape parameter alpha	
17	state frequencies of the assumed stationary distribution	states are bases (DNA) or aa (protein)
18	assumed relative substitution rates	for DNA 6 rates, for protein 190 rates
19	proportion of the assumed rate category	of up to ten rate categories (*+R10*)
20	rate of the assumed rate category	of up to ten rate categories (*+R10*)
21	tree length	
22	sum of internal branch lengths	
23	tree diameter	
24	minimum distance	between pairs of sequences
25	maximum distance	
26	mean distance	
27	median distance	
28	variance in distances	

Cont'd on following page

Table 2.4, cont'd

	feature	**details**
29	minimum branch length	
30	maximum branch length	
31	mean branch length	
32	median branch length	
33	variance in branch lengths	
34	minimum internal branch length	
35	maximum internal branch length	
36	mean internal branch length	
37	median internal branch length	
38	variance in internal branch lengths	
39	minimum external branch length	
40	maximum external branch length	
41	mean external branch length	
42	median external branch length	
43	variance in external branch lengths	
44	Newick string of the tree	

The relevant features gathered for each inferred tree (*initial* or ML) are shown in table 2.4. The features also include the parameter estimates of the model that has been used to construct the tree. Note, that some features are model specific (e.g, a proportion of invariable sites, alpha value for Gamma models) and some only exist for the ML tree and not the *initial tree* (e.g., distances between pairs of sequences calculated based on the tree).

Table 2.5 below depicts the features gathered for each branch. More details, especially regarding the splitsize and path lengths (features 4-11), will be given in section 2.2.2.3.

Table 2.5: Branch features stored in the EvoNAPS database.

	feature	**details**
1	branch type	internal or external
2	branch length	
3	splitsize	
4	minimum path length in subtree 1	calculated as described in section 2.2.2.2
5	maximum path length in subtree 1	
6	mean path length in subtree 1	
7	median path length in subtree 1	
8	minimum path length in subtree 2	
9	maximum path length in subtree 2	
10	mean path length in subtree 2	
11	median path length in subtree 2	

2.2.2 Technical details

To efficiently evaluate a large number of evolutionary models on the alignments gathered from the different sources (see 2.1), a scientific workflow (Taylor et al., 2007) was implemented using the scientific workflow management system Snakemake (v.5.9.1; Köster and Sven, 2012; Mölder et al., 2021). The workflow relies on a customized version of the IQ-Tree2 software (Minh et al., 2020), which will be discussed in detail in section 2.2.2.1 below.

Custom-made Python scripts were used to extract all relevant information and the parameters of interest from the output files produced by IQ-Tree2. The results were written into files especially designed to be imported into the EvoNAPS database. The Python scripts will be discussed in detail in sections 2.2.2.2 and 2.2.2.3. An overview of the workflow is depicted in figure 2.1.

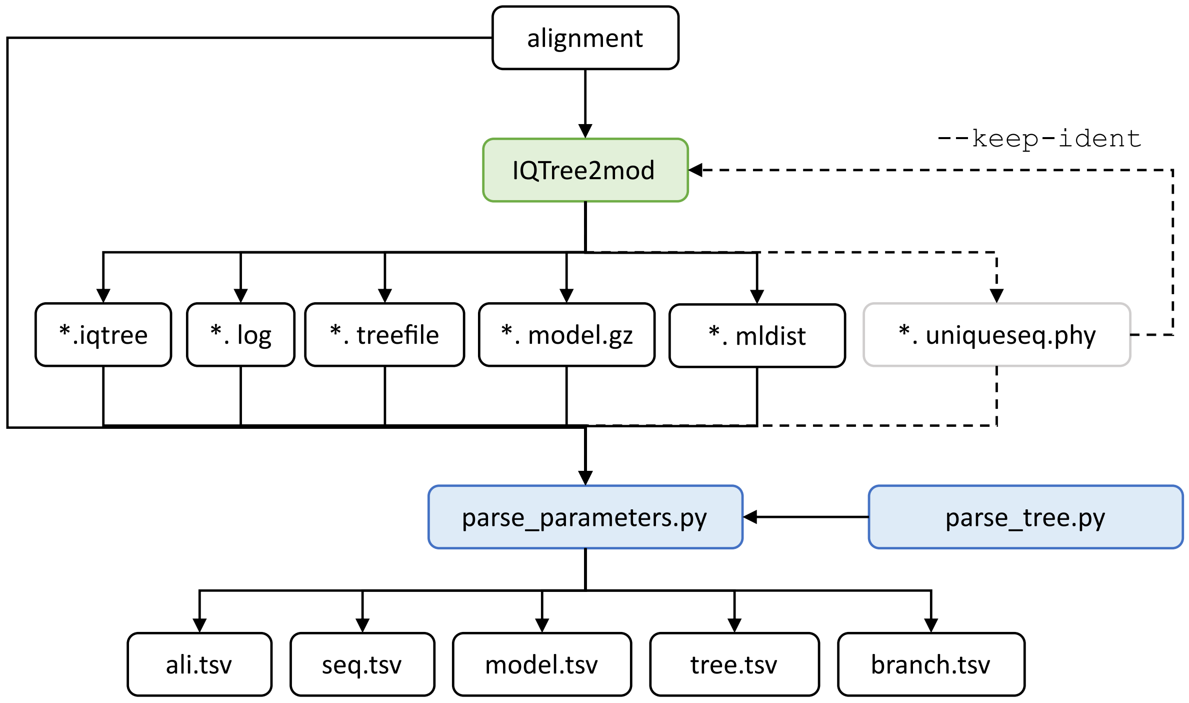

Figure 2.1: The figure schematically depicts the workflow as implemented in Snakemake. An alignment file serves as input (top). First, a modified version of IQ-Tree2 is run on the alignment to both evaluate the different sequence evolution models as well as search for a ML tree based on the best-fit model (according to BIC). Should there be a `uniqueseq.phy` file among the output files generated by IQ-Tree2, then IQ-Tree2 is called again, this time with the additional flag `keep-ident`. The parameters and information that will be imported into the EvoNAPS database is parsed out from the IQ-Tree2 output files and then written into files designed to be imported into the EvoNAPS database using the Python3 script `parse_parameters.py`. Additionally, the `parse_parameters.py` script calls the *parseTree* function from the Python script `parse_tree.py`. The *parseTree* function filters all branch lengths from a tree given in Newick format as well as calculates the paths to the leaves for each internal branch.

2.2.2.1 Model evaluation and tree search

Besides the results of the tree search, the EvoNAPS database also stores the results of model selection. Accordingly, the model performance as well as the parameter estimates of all evaluated models are of interest. However, the "default" version of IQ-Tree 2 (Minh et al., 2020) only returns the parameters of the model that was used in the tree search (or best-fit model). Therefore, the IQ-Tree 2 source code (v. 2.2.0.5) available at `https://github.com/iqtree/iqtree2/releases/tag/v2.2.0.5` was modified accordingly and the parameters of all evaluated models were written into the checkpoint file *`model.gz`. Additionally, the software was modified so that the Newick string and model parameters of the *initial tree* are written into the `*.log` file.

The command line option `-m` (model) as implemented in IQ-Tree2 prompts the user to specify (a) model(s) to be evaluated or to be used in a tree search. In the workflow the `-m` option was set to `MFP`. `MF` stands for *ModelFinder* (as has been described in section 1.2.3). Accordingly, extended model selection is performed on the input alignment that includes *free rate* models. The `P` in `MFP` prompts IQ-Tree2 to immediately follow the model selection with a tree search.

Furthermore, the `--mrate` option was set "`E,I,G,I+G,R`". The `--mrate` option forces IQ-Tree2 to test all declared RHAS models for each substitution rate matrix. Otherwise, a heuristic comes into play that determines early on which RHAS models do not fit the alignment well and excludes them in the remaining model evaluations. The option `E` stands for equal/homogeneous rates model (models assuming no RHAS).

We used the default selection criterion (BIC) to choose the best-fit model. Additionally, the `--seqtype` option was used to manually declare whether the investigated alignment is a DNA or protein alignment.

Apart from the options mentioned, the default settings of IQ-Tree2 were used. For more details regarding the options implemented in IQ-Tree2, please refer to the online manual (Minh et al., 2022).

The substitution rate matrices that are evaluated when using the `MFP` option are the 22 DNA and 16 protein substitution rate matrices as described in table A.2 and table A.1 respectively.

Should a DNA model assume unequal state frequencies, then IQ-Tree2 counts the nucleotides in the alignment and assumes that their relative abundance is equal to the stationary distribution. Models with such empirically estimated state frequencies are marked with a $+F$ flag according to the syntax used by IQ-Tree2.

The 16 protein rate matrices as described in table A.2 come with predefined amino acid frequencies. Either the predefined frequencies are assumed (no addition to the model name), or the state frequencies are empirically estimated ($+F$). Accordingly, there are 32 different protein rate matrices.

As declared in the `--mrate` option, each of the rate matrices was evaluated assuming the same substitution rate(s) across sites ($+E$), an invariable sites model ($+I$), a Gamma model with four discrete rates ($+G4$), a combination of the two ($+I + G4$), or using *free-rate* models ($+Rk$). The *free-rate* models assume at least 2 and at most 10 distribution-free rates k and were evaluated using the modified *ModelFinder* algorithm as described in 1.2.3. Accordingly, each substitution rate matrix is paired with at least two ($+R2$,$+R3$) and at most 9 *free-rate* models ($k = 2...10$).

Overall, at least 132 and at most 286 DNA models and at least 168 and at most 364 protein models were evaluated on each alignment.

To reduce runtime IQ-Tree2 excludes sequences from its calculations should there already exist more than one *identical* sequence in the alignment. Accordingly, IQ-Tree2 keeps exactly two out of each set of identical sequences. In such cases, model selection and the tree search are conducted on the reduced alignment as stored in the *`uniqueseq.phy` file. In the final step of the analysis the excluded sequences are added to the sub-tree in the ML tree where its identical sequences are located. The length of the branch leading to the attached taxon is hereby set to zero ($BL = 0$).

In these cases, for the sake of data integrity we also want to apply the workflow on the original alignment as the exclusion of sequences can influence parameter estimations such as that of the stationary distribution. Accordingly, should an IQ-Tree2 run produce a *`uniqueseq.phy` file, then IQ-Tree2 is applied on the alignment a second time, this time with the `--keep-ident` flag (see figure 2.1). The `--keep-ident` flag ensures that model evaluation and the tree search are conducted on the original alignment that includes all sequences.

The relevant output files generated by IQ-Tree2 include the report file `*.iqtree`, the log file `*.log`, the file with the Newick string of the ML tree `*.treefile`, the file including the distances between each sequence in the alignment calculated based on the best-fit model `*.mldist`, and the checkpoint file `*.model.gz`. Potentially, there is also the alignment file with a reduced number of sequences `*.uniqueseq.phy`.

2.2.2.2 Getting branch lengths and path lengths from a Newick string

The IQ-Tree2 output files described above already contain most of the information we are interested in, especially when it comes to basic information regarding the alignment

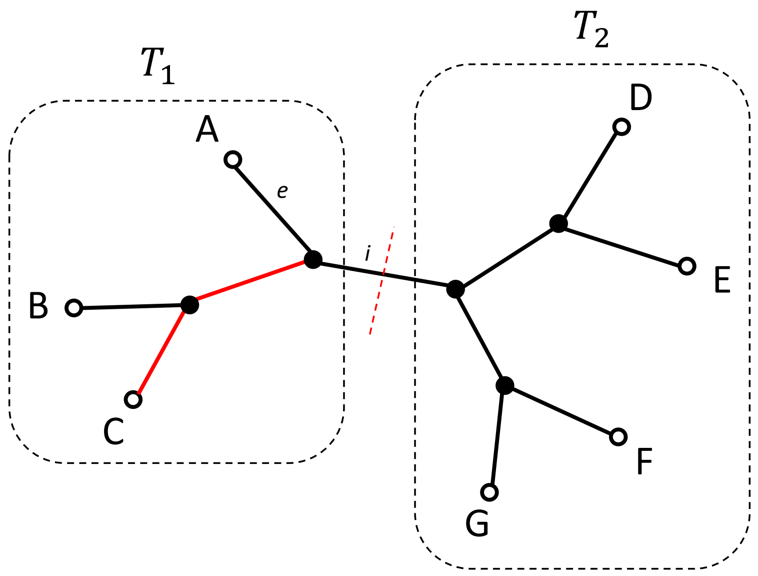

Figure 2.2: A tree T connecting seven taxa A-G is shown. The external nodes (or leaves) are depicted as white dots and the internal nodes as black dots. Branch e is an *external branch* connecting taxon A (an external node) to the tree. Branch i is an internal branch connecting two internal nodes. The red dotted line indicates the split induced by branch i. The split separates tree T into two subtrees T_1 and T_2. The fat red line represents the *path* from node C to branch i. Note, that that branch i itself is not part of the path. The path ends at the earliest encountered node connected to the internal branch i.

and the results of model evaluation (see features in tables 2.1, 2.2, 2.3). However, we are interested in additional parameters especially regarding the resulting phylogenetic trees (*initial tree* or ML tree) that are given in Newick format (Olsen, 1990).

Tree notations. Before proceeding, let us clarify some notations regarding pyhlogenetic trees. Depicted in figure 2.2 is a tree T connecting the seven taxa A-G. The taxa are located at the leaves (or *external nodes*) of the tree (depicted as white dots). A branch connecting a leaf to the tree is called an *external branch* (e.g., branch e). Contrarily, an *internal branch* connects two internal nodes (depicted as black dots; e.g., branch i).

A cut (or split) conducted on any branch on the tree can split the tree into two subtrees. Notably, only splits conducted on internal branches are *non-trivial*. For non-trivial splits both of the resulting subtrees contain more than one taxon as opposed to splits conducted on external branches. As an example, the non-trivial split conducted on branch i in tree T in figure 2.2 cuts the tree into two subtrees T_1 and T_2 (indicated as red dotted line). The *splitsize* is defined as the number of taxa in the smaller subtree (i.e. with less leaves). In our case, the splitsize of branch i is the number of taxa in subtree T_1, which is 3. The splitsize can give an approximation on how deep the branch is located in the tree. However, it does not consider the actual distance (in terms of branch lengths) to the leaves. Here, the calculation of *path lengths* in a tree is necessary.

The path length between any two nodes is defined as the sum of branch lengths of the path connecting the two nodes. As a metric for the depth of a branch in the tree, the path lengths from the leaves to that branch are of particular interest. As an example, in figure 2.2 the path length from the leaf C to the internal branch i is highlighted in red. Note, that the branch length of branch i is not added to the path length. Instead, the lengths of the branches in the path starting from leaf C are summed up until encountering the first internal node connecting to branch i. The *mean path length to the leaves* in the smaller subtree T_1 can be used as metric for the depth of branch i in the tree.

Tree features. Above all, the tree features of interest are the branch lengths. The length of a branch corresponds to the *expected number of mutations per site* along that branch (Bryant et al., 2005). The choice of branch lengths can, therefore, greatly influence the outcome of sequence simulations. Naturally, information whether the branches are internal and external is also relevant. Additionally, the *depth* of a branch in the tree is of interest as it enables the realization of complex simulation scenarios.

Accordingly, two metrics, namely the *splitsize* and the *mean path length to the leaves*, are calculated for each branch in the tree.

Obtaining branch lengths and path lengths. The phylogenetic trees inferred with IQ-Tree2 are given in Newick format (Olsen, 1990). A Python (v.3.3.6) script, `parse_tree.py`, was written to infer branch and tree features from the Newick string. In the script the `SeqIO` and `Phylo` module from the Biopython library (v.1.78; Cock et al., 2009) as well as the NetworkX (v.2.2; Hagberg et al., 2008) and Pandas libraries (v.1.1.3; McKinney, 2010) are used.

The tree in Newick format is read in and interpreted using the `Phylo` module. The tree is then transformed into a undirected graph G with weighted edges using the *Phylo.to_networkx()* function.

It is straightforward to ascertain the branch lengths and the branch type, internal or external, by simply traversing the graph G once. During the graph traversal the weight of each visited edge is evaluated (which coincides with the length of the branch). Additionally, the degree of each visited node is evaluated. An external node has degree 1, an internal node has a degree of at least 3. One graph traversal can be achieved in $O(n)$ time.

Obtaining the path lengths to the leaves for each internal branch is a bit more complicated. An algorithm was implemented to obtain all path lengths in the tree during two depth-first-search post-order graph traversals. This approach follows the idea used in the "Recursive Puzzling Step Algorithm", described in chapter "4.4.2. Recursive Puzzling Step Algorithm" by Schmidt (2003, pp.31-36). The time complexity of the graph traversals is $O(n)$ and that of storing the path lengths during traversal for each branch is also $O(n)$. This yields an overall runtime of $O(n^2)$.

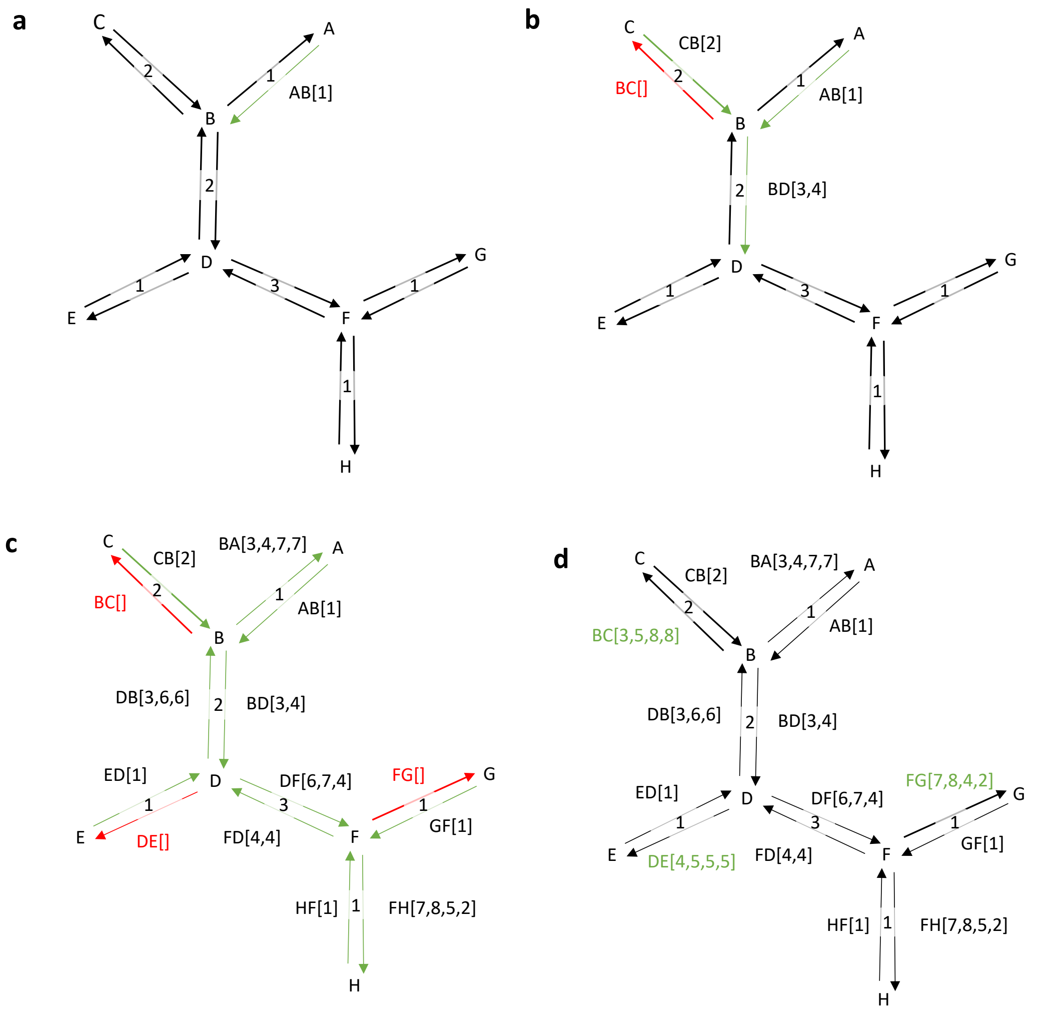

Figure 2.3: The figure depicts four images (**a-d**) of a bi-directed graph representing a fully resolved, unrooted phylogenetic tree. Each directed edge in the graph has the same weight or branch length (represented by the number located at the centre of the edges) in each direction. The edges connect the nodes *A*-*H*. Nodes *A*, *C*, *E*, *G* and *H* only have one neighbouring node and, therefore, represent a leaf or taxon in the phylogenetic tree. In the consecutive figures (**a-d**) an example of how the algorithm that was implemented to calculate all path lengths in a graph is visualized. The graph was traversed in the following order $A \rightarrow B \rightarrow C \rightarrow B \rightarrow D \rightarrow E \rightarrow D \rightarrow F \rightarrow G \rightarrow F \rightarrow H \rightarrow F \rightarrow D \rightarrow D \rightarrow B \rightarrow A$. The first step (**a**) shows the initialisation of the algorithm. Step (**b**) shows how edges *AB*, *CD* and *BD* but not edge *BC* could be *cleared*. Step (**c**) depicts the results from the first graph traversal. Not all edges could be *cleared* as depicted in red. Therefore, a second graph traversal is necessary to *clear* the remaining edges as depicted in (**d**).

In the first step of the algorithm graph G is redefined as a bidirected graph, whereas the weight (or branch length) of edge XY between the nodes X and Y equals the weight of reverse edge YX for any edge in the graph. During graph traversal each directed edge is visited once and the path lengths are consequently calculated by summing up the branch lengths of the traversed edges starting from the leaves.

Let us assume the toy example as depicted in figure 2.3. The graph depicted has 8 nodes *A*-*H*, five of which are leaves. The graph traversal is initiated by visiting node A (see figure 2.3a). The neighbouring node B is visited by traversing edge AB. Because A is a leaf the path length leading from A to B can be calculated, which is the weight of edge $PL_{AB} = |AB| = [1]$. The graph is further traversed by visiting neighbouring node C by traversing edge BC (see 2.3b). In this case not all path lengths leading to C can be calculated. There exit four paths leading to *C*, namely $A \rightarrow C, E \rightarrow C, H \rightarrow C$ and $G \rightarrow C$. So far only the path from the external node A to node C can be calculated, because all the edges in the path $A \rightarrow C$ have already been visited.

More generally, the path lengths to a node Y can be calculated if the edge XY has been *cleared*. An edge XY can only be *cleared* if all path lengths leading to its source node X have already been calculated or, in other words, if all edges leading to X (except reverse edge YX) have also been *cleared*. Accordingly, looking back at the example depicted in figure 2.3b, edge BC cannot yet be *cleared*.

Next, edge *CB* is traversed. Because C is an external node and edge *CB* is therefore the only path leading to the target node *B*, the edge *CB* can be *cleared*. The path length leading to B is the branch length of *B*, $PL_{CB} = |CB| = [2]$. Next node D is visited. All the edges leading to the source node B (except for *DB*) have been *cleared*, namely *AB* and *CB*. Therefore, edge *BD* can also be *cleared*. The path lengths leading to D can be calculated by adding the weight of edge *BD* ($|BD| = 2$) to the calculated path lengths of edges AB ($PL_{AB} = [1]$) and edge CB ($PL_{CB} = [2]$) respectively. The path lengths leading to BD are therefore $PL_{BD} = [3, 4]$.

This procedure is continued until all edges in the graph have been traversed. However, some edges could not be *cleared* in the first graph traversal (depicted in red in figure 2.3c). To *clear* the remaining edges a second graph traversal following the same procedure is necessary. Once the second graph traversal has finished, the algorithm has succeeded in calculating all path lengths leading to each node in the graph (see figure 2.3d).

To summarize, during the graph traversal each node in the graph is visited by choosing a neighbouring node or target node Y of the current node X by traversing the edge XY connecting the two nodes. To ensure that the graph is traversed completely the following restrictions are posed on the decision on which neighbouring node N will become the target node Y:

1. Node N can only become the target node Y if edge XN has not been traversed yet.
2. Always make node N the target node Y if node N has not been visited yet.
3. Should node N have already been visited only make node N the target node if there exist no other neighbouring node that fulfils requirement 2.

The algorithms can be summarized with the following steps:

1. Initialization: make a random node the start node X for traversal.
2. Visit source node X.
3. Find a target node Y that fulfils the requirements as discussed above. Stop the graph traversal if no such node exists anymore.

4. If all edges leading to X (except for edge YX) are *cleared*, calculate the path lengths leading to Y. Mark XY as being *cleared*.
5. Make target node Y the source node X. Go to step 2.

Given the path lengths leading to each node in the graph, statistics regarding the path lengths found in the subtrees separated by an internal branch are now easily calculated. The splitsize equals the number of paths (or leaves) found in the smaller subtree.

Finally, the `parse_tree.py` script writes the branch lengths into a table as well as the information whether the branch is internal or external. Should the branch be internal, then the splitsize, statistics on the path lengths (mean PL, median PL, and the variance in path lengths) of each subtree are stored. Additionally, the script returns characteristics regarding the whole tree such as the tree length, tree diameter, the sum of internal branch lengths as well as statistics describing the distribution of internal and external branch lengths respectively (e.g., mean internal BL, mean external BL).

2.2.2.3 Feature extraction

We now have obtained features of the alignment, evaluated models and inferred trees as stored in the IQ-Tree2 output files (section 2.2.2.1) and can infer additional parameters for the generated trees using the `parse_tree.py` script (section 2.2.2.2).

Next, the features need to be organised in a way that they can easily be imported into the EvoNAPS database. For this purpose, the script `parse_parameters.py` written in Python (v.3.3.6) was used. The script relies on the data analysis library Pandas (v.1.1.3; McKinney, 2010) and uses the SeqIO module from the Biopython library (v.1.78; Cock et al., 2009) to read in the sequences from the alignment.

The necessary input for the `parse_parameters.py` script are the original alignment file, the output files generated by IQ-Tree2 and the `parse_tree.py` script. The output are a total of five files in tab separated values (`tsv`) format. Each file corresponds to a table in the EvoNAPS database, whereas the column names in the `*.tsv` files coincide with that of the EvoNAPS tables respectively.

2.3 Database architecture

The EvoNAPS database should contain the original data sets (alignments) as well as the results of the model evaluations and tree search. Hence, the sequences, evaluated models and inferred trees need to be clearly assigned to any given alignment. Additionally, the origin (see section 2.1) of the alignment should be clear. Therefore, a *relational* and mostly *hierarchical* design of the EvoNAPS database is necessary. EvoNAPS was implemented using the open-source MariaDB Server (v.10.0.13).

Overall, the EvoNAPS database is comprised of 13 tables, whereas in each table auto-incremented integers serve as *primary keys*. For each table containing information of DNA alignments there exists a corresponding table for protein alignments. The tables are set in relation to each other by employing *foreign keys*. Data integrity is usually kept by *constraining* a set of columns in the tables to form a *unique key*. An overview of the database can be viewed in figure 2.4. The figure shows an Entity-relationship (EER) diagram of EvoNAPS, whereas only primary (yellow) and foreign keys (red) as well as columns involved in unique key constraints (blue) are depicted.

In the following, the tables in the EvoNAPS database will be discussed in more detail especially in regards to what information each table contains and how they relate to each other. The exact number of columns and column names, though not discussed here, can be viewed in the tables in the appendix (see B).

The *dataorigin* table. At the top of the EvoNAPS database stands the *dataorigin* table, which contains information regarding the origin of each alignment in EvoNAPS, namely PANDIT, Lanfear or OrthoMaM (see section 2.1). Each source database described in the *dataorigin* table has a unique name `DATABASE_ID` corresponding to the name of the original database. The *dataorigin* table is connected to the *alignments* tables using the `DATABASE_ID` in *dataorigin* as foreign key that connects to the `FROM_DATABASE` column in *alignments*. For more details, refer to section B.1 in the appendix.

The *alignments* tables contain information regarding each alignment stored in the database such as the number of sequences, sites, and patterns in the alignments. The name of the alignment, `ALI_ID`, must be unique (constraint). The column `ALI_ID` serves as foreign key in the *sequences*, *modelparameters* and *trees* tables. (See sections B.4 and B.5 for more details.)

The *sequences* tables contain the sequences of each alignment stored in the *alignments* tables. Each sequence has an index `SEQ_INDEX` starting with 1. Paired with the `ALI_ID` column the `SEQ_INDEX` column forms a constraint (unique key) on the sequences table. Accordingly, there can only be one sequence with index i for the same alignment. This ensures that there can be multiple sequences for one alignment but prevents the possible error that the same sequences are added to the database multiple times. Besides the sequences themselves, information such as the nucleotide or amino acid composition of the sequence are stored in the table. (See sections B.6 and B.7 for more details.)

The *modelparameters* tables. The estimated parameters for the sequence evolution models that were evaluated on the alignments in the *alignments* table are stored in the *modelparameters* table. The two tables are connected using the `ALI_ID` column. Each row in *modelparameters* includes the parameters of one evaluated sequence evolution model such as the equilibrium state frequencies `STAT_FREQ_*`, the substitution

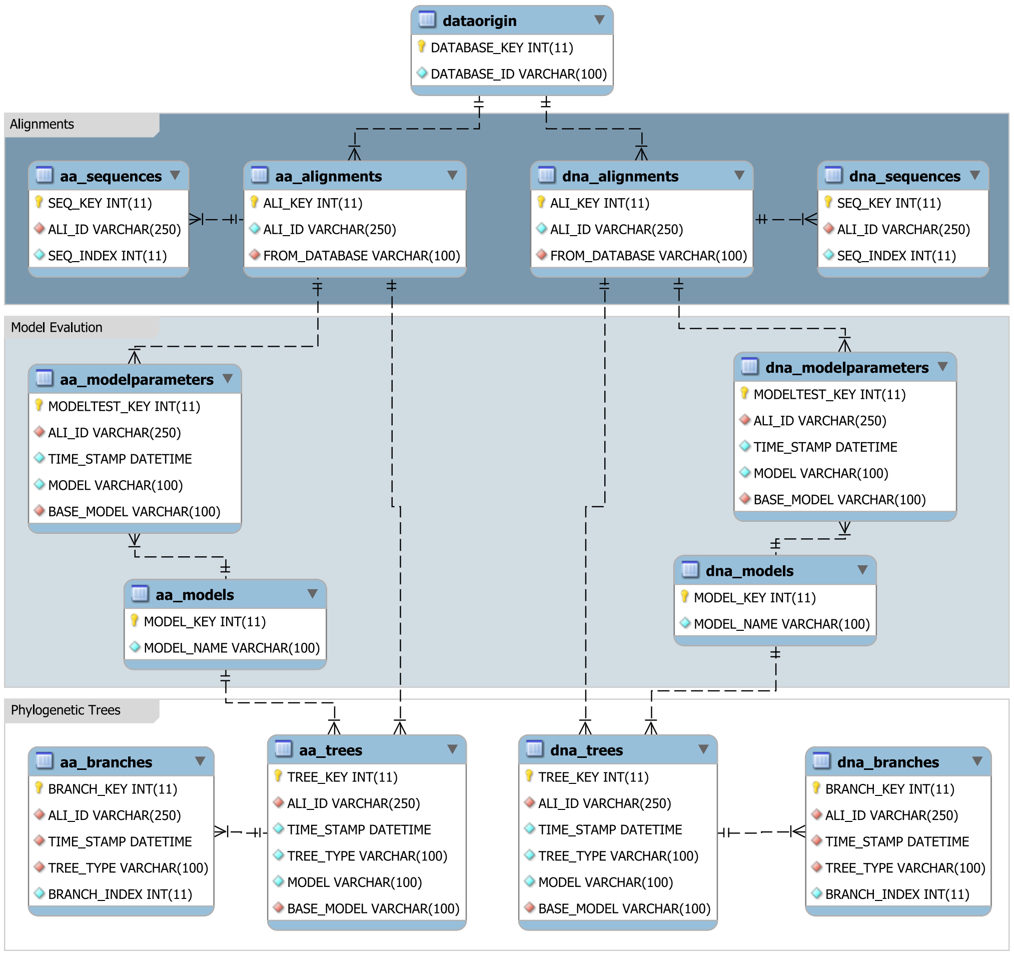

Figure 2.4: The figure shows an enhanced Entity-relationship (EER) diagram of the EvoNAPS database using Crow's feet notation. For simplicity only primary and foreign keys as well as columns involved in constraints are depicted. The database is symmetric in the sense that, except for the *dataorigin* table, for each table containing parameters and information of DNA alignments there exists a corresponding table for protein alignments. The relational design of the database is depicted with the dotted lines connecting the tables. Primary keys are depicted with a yellow symbol. If a column is part of a constraint, it is depicted in blue. A column serving as a foreign key is depicted with a red symbol. The figure was created using the reverse engineering tools of the MySQL Workbench program available at `https://www.mysql.com/products/workbench/`.

rates `RATE_*`, possibly the shape parameter alpha `ALPHA`, the proportion of invariable sites `PROP_INVAR`, and so on. Additionally, information regarding how the model performed is provided, such as the logarithmic likelihood (`LOGL`) and also the values for `AIC`, `BIC`, `AICc` as well as their calculated weights. There exists a constraint on the *modelparameters* table that forces the `ALI_ID`, `MODEL_NAME` and `TIME_STAMP` columns to form a unique key. This constraint ensures that the same results are not added multiple times and/or that the data is not overwritten by possible future data imports. The timestamp `TIME_STAMP` is hereby taken from the `*.iqtree` output file. This ensures that each model `MODEL_NAME` that was evaluated on alignment `ALI_ID` in one IQ-Tree2 run (given by `TIME_STAMP`) can only be added once. Furthermore, the column `IQTREE_VERSION`, which states the version of IQ-Tree that has been used to evaluate the model, and the random number seed stored in the `RANDOM_SEED` column, ensure that an exact reproduction of the data is possible. The `KEEP_IDENT` column states whether IQ-Tree was run with the `--keep-ident` option (1) or not (0). (See sections B.8 and B.9 for more details.)

The *trees* tables. For each alignment there are at least two trees stored in the *trees* table: the *initial tree* T used for the evaluation of the candidate models and the ML tree. Besides the trees, also the name of the model and the model parameter estimates that were used for tree reconstruction are included. In case of DNA alignments, the *initial tree* was always inferred using model *GTR+I+G4*, in case of AA alignments model *LG+I+G4* was used. The ML tree was inferred using the best-fit model M that was determined during model selection. In comparison to the model parameters as stored in the *modelparameters* tables, the parameters of the best-fit model M were estimated based on the ML tree (and not the *initial tree*). Furthermore, the parameters of model M were optimized on the ML tree in a step following the ML tree search.

For each tree in the *trees* tables the NEWICK_STRING, TREE_LENGTH, the TREE_DIAMETER, and statistics regarding the branch length, e.g., MEAN_BL, are stored. The columns ALI_ID, TIME_STAMP and TREE_TYPE form a unique key constraint to ensure data integrity. The column TREE_TYPE is simply a flag stating whether the concerned tree is a ML tree or initial tree. Accordingly, there can at most be two trees with the same ALI_ID and TIME_STAMP. The columns IQTREE_VERSION, KEEP_IDENT and RANDOM_SEED ensure data reproducibility. (See sections B.10 and B.11 for more details.)

The *branches* tables. Each branch length (BL) in a tree is stored in the *branches* table and is connected to its corresponding tree in the *trees* table via a foreign key formed by ALI_ID, TIME_STAMP and TREE_TYPE. Additionally, the columns forming the foreign key are paired with the BRANCH_INDEX, starting at 1, to form a unique key constraint on the table. The constraint ensures that the same branches are not added multiple times. The branches table include the branch lengths, characteristics regarding the branch such as whether it is internal or external, its splitsize and statistics regarding path lengths, e.g., MEAN_PATH. Note that if the branch is external, then the leaf connected to the branch coincides with the sequence stored in the sequences table where SEQ_INDEX equals the BRANCH_INDEX. (See sections B.12 and B.13 for more details.)

The *models* tables in the database gives a short description and overview of the parameters of the different substitution rates matrices. The table is connected to the *trees* and *modelparameters* table using the unique MODEL_NAME column in the models table that serves as foreign key connecting to the BASE_MODEL column in both the *trees* and *modelparameters* tables. (See sections B.2 and B.3 for more details.)

Overall, each alignment stored in the alignments table can be linked to its source database as described in the *dataorigin* table. The sequences as they appear in the original alignment are stored in the *sequences* table. The parameters of each sequence evolution model evaluated on the alignment are stored in the *modelparameters* table. The *initial tree* that was used for model evaluation as well as the inferred ML tree are stored in the *trees* table. For each tree the branch lengths as well statistics regarding its branches (e.g., splitsize, mean PL) are stored in the *branches* table.

Chapter 3

Results

The workflow that was introduced in section 2.2 was applied to the alignments from the three sources that were described in section 2.1. Overall, 22,678 DNA alignments and 6,614 AA alignments were analysed. For each alignment the best-fit model (according to BIC) was determined in a model selection step, which was then used in a ML tree search. Overall, the model parameters of over 4.2 million evaluated DNA models and over 2.5 million evaluated protein models were gathered. Furthermore, over 32,000 ML trees were inferred from the alignments. Table 3.1 gives an overview of the size of the EvoNAPS database. Notably, some of the trees and evaluated models stem from additional IQ-Tree2 runs with enabled `--keep-ident` flag.

The EvoNAPS database comes equipped with various filter options that the user can employ to search for data that closely matches the one they want to simulate. The filter options can be classified according to the tables in the EvoNAPS database that are being searched.

The *alignments* table can be used to filter for alignments that stem from a specific source (using the `FROM_DATABASE` column). Furthermore, one can restrict the search

Table 3.1: Size of the EvoNAPS database. The keep-ident column states whether the corresponding numbers were produced by IQ-Tree2 runs with (+) or without (-) the `--keep-ident` option.

	keep-ident	**DNA**	**Protein**
Alignments		22,678	6,614
Sequences		1,893,846	175,287
Model evaluations	-	3,817,531	2,548,436
ML trees	-	22,678	6,614
***initial* trees**	-	22,678	6,614
Model evaluations	+	456,206	16,041
ML trees	+	2,769	40
***initial* trees**	+	2,769	40
BLs		8,539,464	672,738

to alignments with a certain number of sequences, sites, patterns and/or parsimony informative sites.

The *modelparameters* table can be searched for different evolutionary models of interest and how well they performed in comparison to each other using, e.g., their Akaike or Schwarz weights (Burnham and Anderson, 2001; Wagenmakers and Farrell, 2004).

The trees stored in the *trees* table can be filtered regarding their overall length, diameter, the sum of internal BLs and/or mean BLs. Additionally, the parameter estimates of the best-fit model are also included in the table.

The remainder of this chapter will give examples on how to use the provided features and filter options to search for alignments, models parameter estimates and/or trees stored in the EvoNAPS database.

3.1 Extracting model parameter estimates

In this example we want to obtain model parameter estimates of the DNA sequences evolution model *GTR+I+G* to use them as input for sequence simulations. The parameters estimates of interest are the equilibrium state frequencies, the relative substitution rates, the proportion of invariable sites (*+I*) and the shape parameter alpha. Furthermore, we want to restrict the search to model parameters that were estimated on alignments that contain at least 10 sequences and with at least 100 aligned sites.

First of all, we are presented with two options, whether to search for model *GTR+I+G* in the *dna_modelparameters* or the *dna_trees* table. Both tables include parameter setting of evolutionary models that were estimated based on biological data. However, the model parameters in the *dna_modelparameters* table were calculated during model selection and were estimated based on the *initial tree*. In comparison, the *dna_trees* table includes the parameter settings of the best-fit model that was used for the ML tree search. Additionally, the model parameters were further optimized on the ML tree in a final step during the pyhlogenetic analysis.

Accordingly, the advantage of searching in the *dna_trees* table is that we always obtain optimized parameters. However, the number of available parameter settings in the *dna_trees* table is considerably smaller than in the *dna_modelparameters* table (see table 3.1). Should the user decide to search the *dna_modelparameters* table to obtain a larger number of and more diverse parameter settings, we recommend restricting the search to models that fit the alignment considerably well using, e.g., their "Schwarz" weights (see chapter 1.2.2; Burnham and Anderson, 2001).

Independent on which table is chosen for the search, in both cases features found in the *dna_alignments* table are used to filter the results. These features are the number of

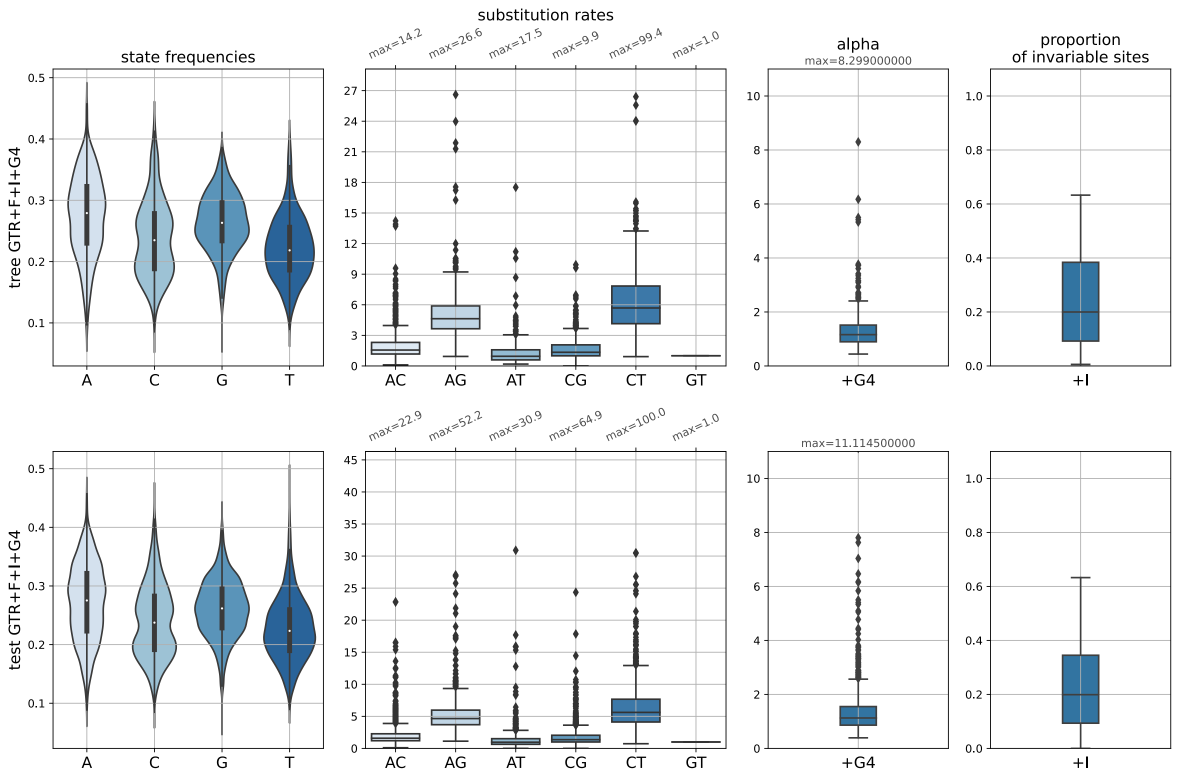

Figure 3.1: The figure depicts the distribution of the parameters of the model *GTR+I+G4* exported from the EvoNAPS database after restricting the search to models that were evaluated on alignments with at least 10 sequences and at least 100 sites. On the top the parameter distributions that were exported from the *dna_trees* table are visualized. Overall, 723 trees were found with the given restrictions. The search in the *dna_modelparameters* table was further restricted to only output result of models that fit the data considerably well ($wBIC > 0.5$). Overall, 2048 evaluated models that satisfied the restrictions were found in the *dna_modelparameters* table. The resulting parameter distribution are depicted on the bottom.

sequences (≥ 10) and sites (≥ 100) in the alignment the model parameter estimations are based on. Accordingly, the query created to fetch the model parameter estimates joins the *dna_alignments* table with either the *dna_trees* or the *dna_modelparameters* table.

In our example, the search in the *dna_trees* table results in 723 parameter settings of model *GTR+I+G4*. The search in the *dna_modelparameters* table is restricted to models with a "Schwarz" weight of $wBIC \geq 0.5$, which results in a total of 2047 hits. The distributions of the fetched model parameters are displayed in figure 3.1.

The obtained model parameter settings can directly be used as input for sequence simulations. Accordingly, the number of different parameter settings coincides with the number of hits in the database. However, sometimes there is a need for more diverse and a greater number of parameter settings, e.g., for the generation of training data for machine learning algorithms. In these cases, parameters can be sampled from the individual parameter distributions (as depicted in figure 3.1) as opposed to the joint distributions. Hence, the number of possible parameter settings is increased by the number of possible combinations of the individual parameters. Note, however, that with individual parameter sampling the dependencies between parameters are being disregarded. Thus, while the individual parameter estimates are still based on biological data, the combination of the sampled parameters might be unrealistic.

3.2 Extracting branch lengths

The choice of branch lengths can greatly influence the outcome of sequence simulation as they represent the mean number of mutations per site along that branch. The *branches* tables in the EvoNAPS database provide the user with branch lengths estimates based on empirical alignments.

Table 3.2: The table shows the number of hits in the EvoNAPS database obtained in the search for branch length estimates of ML trees containing 60-80 taxa. The results are broken down according to the source of the alignments the found trees are based on (source) as well as the type of branch (internal or external) of the obtained branch lengths.

source	**#trees**	**#branches**	**type**	**mean BL**	**max BL**
DNA_Lanfear	246	15,912	internal	0.03193	4.45185
		16,650	external	0.10169	10.17519
DNA_OrthoMaM	518	35,832	internal	0.02204	1.20669
		37,386	external	0.03701	1.10516
DNA_PANDIT	190	12,393	internal	0.18799	7.96281
		12,963	external	0.45778	13.01255
AA_PANDIT	208	13,680	internal	0.28576	13.20629
		14,304	external	0.58002	10.18376

In our example we want to sample empirical branch length estimates for a tree containing 60-80 taxa and restrict the search to ML trees. Overall, EvoNAPS holds 1,162 trees that fit the requirements, resulting in 159,120 branch length estimates to sample from. The majority of trees were inferred based on DNA alignments from the OrthoMaM database (518 trees), followed by DNA alignments provided by Rob Lanfear (246) and AA alignments (208) and DNA alignments (190) from the PANDIT database (see table 3.2).

Figure 3.2 shows how the branch lengths are distributed. Evidently, the external branches tend to be longer than the internal branches. Furthermore, branch length estimates based on alignments from the PANDIT database tend to be considerably higher than those of the Lanfear and OrthoMaM databases. This difference in branch lengths reflect the general evolutionary distance between the sequences in the alignments, which are only mammalian in the OrthoMaM alignments, but can stem from all domains in life in the PANDIT alignments.

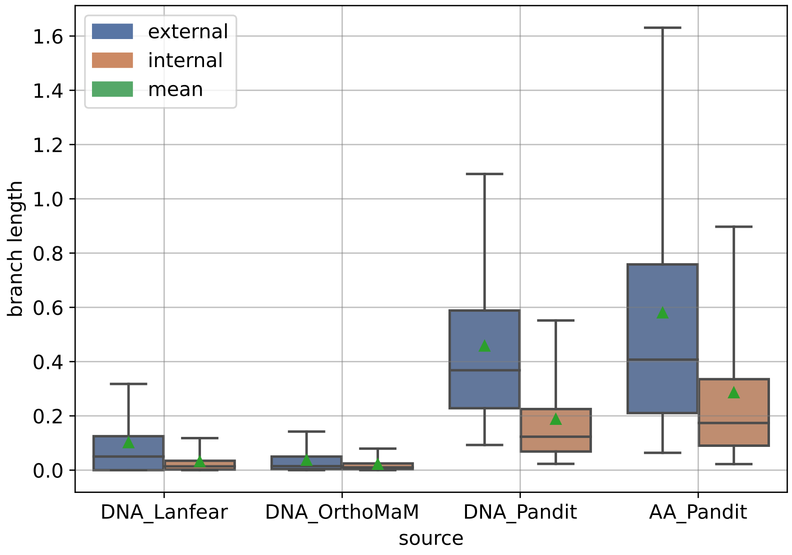

Figure 3.2: Branch length distributions of trees with 60-80 taxa that were inferred based on alignments from the different sources Lanfear, OrthoMaM and PANDIT. The PANDIT data sets are further split into DNA and AA alignments (Lanfear and OrthoMaM alignments are all DNA alignments). The boxplots show how the branch lengths are distributed. The whiskers mark the 5th and 95th percentile. Outliers were excluded for clarity. The distributions of branch lengths of external branches are depicted in blue, those of internal branches in orange. The mean branch lengths are indicated with a green triangle. The overall number of branch lengths as well as the maximum values for the branch lengths for each data source can be viewed in table 3.2.

There are additional filter options should the user wish to also consider the evolutionary distances between the taxa in the tree. These options are either the tree diameter (which indicates the maximum path length between two taxa in the tree) or the pairwise distances between sequences that were calculated using the best-fit model (see features 23-28 in table 2.4 on page 28).

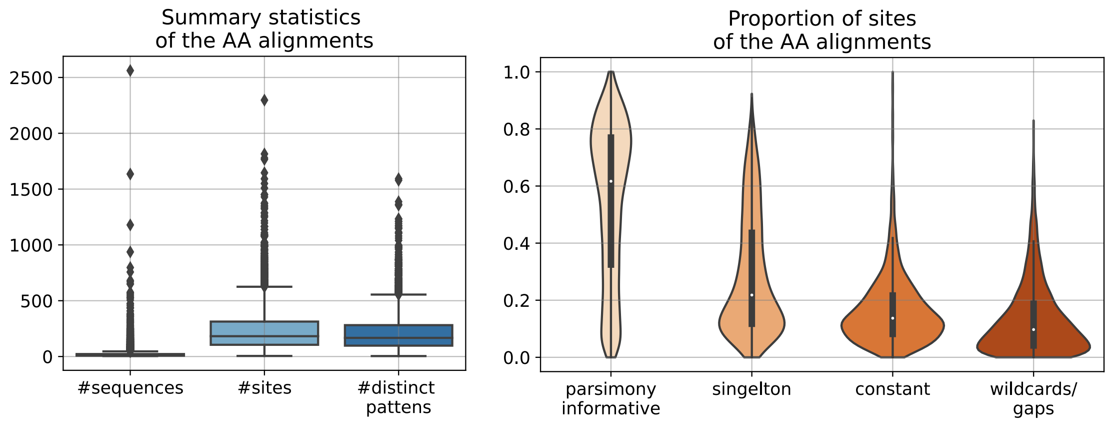

Figure 3.3: The figure shows how the summary statistics of the AA alignments in the EvoNAPS database are distributed. The distributions include the summary statistics of overall 6,654 AA alignments.

3.3 Extracting summary statistics

In this example, we want to extract the summary statistics of all protein alignments found in the EvoNAPS database. So far, EvoNAPS holds 6,654 protein alignments, all of which originate from the PANDIT database. Figure 3.3 shows the summary statistics of the alignments, which include the number of sequences, columns and distinct patterns as well as the proportion of sites that are parsimony informative, singleton, constant or wildcards/gaps. While the maximum number of sequences in an alignment is 2,562, the median number of sequences is 11.

3.4 Obtaining results of model selection

In this example, we want to obtain the results of model selection. For this purpose the best-fit models from the *trees* tables were extracted. The two tables 3.3 and 3.4 show how often DNA and protein model respectively were found to be the best-fit model (according to BIC). The columns of the tables depict the RHAS, which are ordered according to their number of free parameters from left to right. The substitution rate

Table 3.3: The table shows the results of model selection by depicting how often which DNA model was chosen as best-fit model (according to BIC). The substitution models are ordered top to bottom according to how often they were found to be the best-fit model. The RHAS models are ordered left to right according to the number of free parameters of the model.

	uniform	+I	+G4	+I+G4	+R2	+R3	+R4	+R5	+R6	+R7	+R8	+R9	+R10	sum	%
F81+F	22	26	19	4	6	0	0	0	0	0	0	0	0	77	0.30
JC	39	19	34	1	9	0	0	0	0	0	0	0	0	102	0.40
TIMe	0	2	11	72	2	8	32	16	3	1	0	0	0	147	0.58
TIM2e	1	4	36	99	3	12	38	12	0	0	0	0	0	205	0.81
TNe	5	11	56	103	9	35	34	8	3	0	0	0	0	264	1.04
K3P	3	14	64	167	10	34	63	23	12	2	0	0	0	392	1.54
K3Pu+F	6	16	40	132	9	35	135	30	8	4	1	0	0	416	1.63
TPM2	6	24	224	284	15	41	84	16	8	0	0	0	0	702	2.76
TIM+F	1	9	38	211	3	51	361	124	20	6	0	0	0	824	3.24
HKY+F	45	104	256	200	23	56	173	23	1	0	0	1	1	883	3.47
TIM3e	1	0	63	405	7	65	334	87	10	1	1	0	0	974	3.83
TN+F	4	37	93	313	15	80	362	100	16	0	0	0	0	1020	4.01
TPM2u+F	7	33	129	315	15	46	386	84	15	1	0	0	0	1031	4.05
SYM	1	1	26	315	8	52	430	173	65	13	6	0	5	1095	4.30
K2P	48	133	469	327	46	96	81	17	2	1	0	0	0	1220	4.79
TIM2+F	4	6	53	449	9	62	566	155	37	6	2	1	2	1352	5.31
TVM+F	5	11	206	569	49	180	695	209	58	18	6	3	1	2010	7.90
TVMe	2	10	118	721	44	163	832	204	66	11	4	1	1	2177	8.56
TIM3+F	4	16	258	751	10	172	787	242	46	12	3	1	0	2302	9.05
TPM3u+F	45	106	868	661	79	142	405	84	6	0	1	0	0	2397	9.42
TPM3	28	64	596	895	44	186	735	54	10	1	1	1	0	2615	10.28
GTR+F	2	3	160	733	11	166	1302	618	185	36	14	7	5	3242	12.74
sum	279	649	3817	7727	426	1682	7835	2279	571	113	39	15	15	25447	
%	1.10	2.55	15.00	30.37	1.67	6.61	30.79	8.96	2.24	0.44	0.15	0.06	0.06		

matrices depicted in the rows are ordered according to how often they were part of the best-fit model.

DNA models. Table 3.3 shows the results of model selection of the DNA alignments by depicting how often which model was found to be best-fit model. The substitution rate matrix most often found to be part of the best-fit model is the *GTR* rate matrix comprising 3,242 of the 25,447 cases (12.74%). The RHAS model that was most frequently part of best-fit model is the *free-rate* model with four free rates (*+R4*) and accounts for 7,835 cases (30.79%). The Gamma model paired with a proportion of in-

Table 3.4: The table shows the results of model selection by depicting how often which protein model was chosen as best-fit model (according to BIC). The substitution models are ordered top to bottom according to how often they were found to be the best-fit model. The RHAS models are ordered left to right according to the number of free parameters of the model

	uniform	+I	+G4	+I+G4	+R2	+R3	+R4	+R5	+R6	+R7	+R8	+R9	+R10	sum	%
LG	21	37	991	386	54	57	34	45	23	16	2	0	1	1667	25.05
Q.pfam	13	31	624	247	52	41	48	51	42	16	9	0	3	1177	17.69
Q.yeast	17	15	288	70	29	4	8	4	1	1	0	0	0	437	6.57
WAG	40	39	195	85	27	20	11	6	3	1	0	0	0	427	6.42
Q.insect	15	22	231	48	13	4	2	6	0	3	0	0	0	344	5.17
LG+F	0	0	163	68	7	10	12	16	6	7	2	0	0	291	4.37
Q.pfam+F	1	4	103	55	9	24	29	23	13	9	2	0	2	274	4.12
VT	36	31	93	35	19	16	14	12	4	0	1	0	0	261	3.92
Blosum62	18	14	45	25	11	11	5	5	1	1	0	0	0	136	2.04
PMB	28	13	47	12	9	3	3	1	2	0	0	0	1	119	1.79
Q.plant	4	5	85	20	1	1	1	0	1	0	0	0	0	118	1.77
Q.mammal	20	20	61	7	7	1	0	0	0	0	0	0	0	116	1.74
WAG+F	5	7	39	22	6	12	11	8	2	0	0	1	0	113	1.70
JTT	12	7	76	10	6	0	0	1	0	0	0	0	0	112	1.68
VT+F	2	3	37	13	6	10	8	7	4	3	2	0	0	95	1.43
FLU	10	4	66	5	6	2	1	0	0	0	0	0	0	94	1.41
mtZOA	3	1	69	3	7	0	2	1	1	0	0	0	0	87	1.31
Q.yeast+F	0	1	48	14	3	2	3	0	2	0	1	0	0	74	1.11
Q.insect+F	3	3	43	12	2	0	1	0	0	0	0	0	0	64	0.96
rtREV	1	7	42	6	4	1	0	1	0	0	0	0	0	62	0.93
JTTDCMut	5	10	34	9	2	0	0	0	0	0	0	0	0	60	0.90
cpREV	7	2	30	4	4	0	0	1	0	0	0	0	0	48	0.72
Q.bird	14	8	20	2	0	0	0	0	0	0	0	0	0	44	0.66
Blosum62+F	4	1	10	7	2	5	4	2	1	1	2	0	0	39	0.59
DCMut	5	6	23	2	0	0	1	0	0	0	0	0	0	37	0.56
JTT+F	0	1	19	5	1	1	4	2	0	0	0	0	0	33	0.50
PMB+F	0	0	17	3	2	2	5	1	0	1	1	0	1	33	0.50
Q.mammal+F	1	2	26	3	0	1	0	0	0	0	0	0	0	33	0.50
Dayhoff	6	7	17	2	0	0	0	0	0	0	0	0	0	32	0.48
mtInv+F	1	0	22	3	1	2	0	1	1	1	0	0	0	32	0.48
rtREV+F	2	0	20	3	0	0	1	1	1	0	0	0	0	28	0.42
FLU+F	1	2	19	1	1	2	0	0	0	0	0	0	0	26	0.39
cpREV+F	0	2	11	2	0	0	1	0	0	0	0	0	0	16	0.24
HIVb	5	2	8	1	0	0	0	0	0	0	0	0	0	16	0.24
Q.plant+F	0	0	11	2	1	2	0	0	0	0	0	0	0	16	0.24
JTTDCMut+F	2	0	10	2	0	0	0	0	0	0	0	0	0	14	0.21
Q.bird+F	3	3	5	1	0	0	0	0	0	0	0	0	0	12	0.18
HIVw	5	1	4	0	0	0	0	0	0	0	0	0	0	10	0.15
FLAVI	2	3	4	0	0	0	0	0	0	0	0	0	0	9	0.14
Dayhoff+F	1	2	2	1	0	0	0	0	0	0	0	0	0	6	0.09
mtMet	0	1	3	1	0	0	0	1	0	0	0	0	0	6	0.09
HIVb+F	1	0	4	0	0	0	0	0	0	0	0	0	0	5	0.08
HIVw+F	2	1	2	0	0	0	0	0	0	0	0	0	0	5	0.08
mtInv	2	0	3	0	0	0	0	0	0	0	0	0	0	5	0.08
mtZOA+F	0	0	4	0	0	0	0	0	0	0	0	0	0	4	0.06
DCMut+F	0	0	3	0	0	0	0	0	0	0	0	0	0	3	0.05
mtMet+F	2	0	1	0	0	0	0	0	0	0	0	0	0	3	0.05
mtREV	0	0	2	1	0	0	0	0	0	0	0	0	0	3	0.05
mtVer	1	1	0	1	0	0	0	0	0	0	0	0	0	3	0.05
FLAVI+F	1	0	1	0	0	0	0	0	0	0	0	0	0	2	0.03
mtREV+F	0	0	2	0	0	0	0	0	0	0	0	0	0	2	0.03
mtMAM	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0.02
sum	322	320	3683	1199	292	234	209	196	108	60	22	1	8	6654	
%	4.84	4.81	55.35	18.02	4.39	3.52	3.14	2.95	1.62	0.90	0.33	0.02	0.12		

variable sites (*+I+G4*) comes second by a small margin counting 7,727 cases (30.37%). Overall, the model most often found to be the best-fit model is the *GTR+R4* model with 1,302 cases (5.12%).

AA models. Table 3.4 shows how often which protein sequence evolution model was found to be best-fit model (according to BIC). The model most often found to be the best-fit model is *LG+I+G4* with 991 out of 6,654 cases (14.89%). The rate matrix *LG* was part of the best-fit model in 25.05% of cases. The RHAS model *+G4* was part of the best-fit model in 55.35% of cases and model *+G4+I* in 18.02% of cases. Rate matrices assuming the predefined equilibrium state frequencies of the matrix were part of the best-fit model in 81.62% of cases as compared to using the empirical estimation of frequencies (*+F*) with 18.38%.

Note, that the displayed numbers in tables 3.3 and 3.4 also include the results of model selection of IQ-Tree2 runs with `--keep-ident` flag. Hence, the number of conducted model selections is greater than the number of alignments in the EvoNPAS database.

Chapter 4

Discussion

The EvoNAPS database holds parameter settings of numerous evolutionary models and phylogenetic trees that are based on biological data. EvoNAPS allows the user to explore how model and tree parameters are distributed in biological data in different contexts by using the large number of available features as search filters. The last chapter gave some examples on what features can be extracted and/or can be used for filtering (sections 3.1-3.4).

So far, EvoNAPS holds data resulting from analysing alignments provided by Rob Lanfear (Lanfear, 2019) and that of the OrthoMaM (Douzery et al., 2014) and PANDIT (Whelan et al., 2006) databases. Accordingly, the gathered features stored in EvoNAPS are biased. The majority of DNA alignments originate from the OrthoMaM database and, accordingly, hold strictly mammalian sequences. Protein alignments are still underrepresented and consist of primarily small alignments (in regards to the number of sequences; see figure 3.3). Hence, an expansion of the database with additional alignments is required to make EvoNAPS more representative. The expansion can easily be achieved using the custom-made workflow as described in chapter 2.2).

Notably, there exists a database similar in spirit to the EvoNAPS database, namely the RAxML Grove database (Höhler et al., 2022). RAxML Grove provides over 60,000 phylogenetic trees and the respective model parameters estimates. They are based on anonymized empirical alignments that were analysed using the RAxML (Stamatakis, 2014) or RAxML-NG (Kozlov et al., 2019) inference software on two web servers. Like EvoNAPS, the RAxML Grove database was created to provide input for sequence simulations that is based on empirical data.

However, there exist some clear difference between the two databases especially regarding how the data that they hold was generated. For EvoNAPS we followed a typical phylogenetic inference workflow that includes a model selection step as well as a tree search using the best-fit model. The RAxML and RAxML-NG webserver offer the same choice of models as the ones that were tested in our workflow (see section 2.2 for details). However, the workflow provided on the two webserver does not include model selection based on the alignment. Instead, the user can choose a model themselves. Hence, the choice often falls onto popular models, but that do not necessarily fit the alignment well. For instance, for DNA alignments the popular GTR rate matrix (Tavaré, 1986) was chosen in 99% of cases, whereas the remaining rate matrices are underrepresented.

In comparison, the trees in the EvoNAPS database were inferred using the best-fit model (according to BIC). As a result, the models used for the ML tree search are more diverse as compared to those in RAxML Grove (see table 3.3). For example, DNA models with the rate matrix GTR comprise 12.74% of cases in EvoNAPS (as compared to 99% in RAxML Grove).

Additionally, besides the results of the tree search, also the detailed results of model selection are included in the EvoNAPS database (as stored in the *modelparameters* tables). Therefore, it is transparent how well the models fit the alignment in comparison to each other. Furthermore, the model parameters estimates of the evaluated models can also be used as input for sequence simulations.

Another fundamental difference between the two databases is that in RAxML Grove the alignments that have been used for parameter estimation were provided by the users of the RAxML and RAxML-NG webserver and, hence, were anonymized. Checking the original data sets and reproducing the results is, therefore, impossible. In comparison, the alignments that have been used to create the data for EvoNAPS are known and available as they were taken from published sources (see section 2.1).

Chapter 5

Conclusion and Outlook

Phylogenetic studies heavily rely on simulated data. However, it is often unclear how the choose the input for sequence simulations. For this purpose the EvoNAPS database was built. It is a database providing parameter estimates of numerous models of sequence evolution as well as phylogenetic trees that are based on biological data.

To built the EvoNAPS database, biological alignments from published sources were gathered and analysed by applying a typical model-based phylogenetic workflow. The workflow includes a model selection step with subsequent ML tree search and relies on the phylogenetic inference software IQ-Tree2 (Minh et al., 2020) as well as custom-made Phyton scripts. The alignments and inferred trees as well as corresponding model parameter estimates were extracted from the results and stored in the EvoNAPS database. Additionally, various features regarding the alignment, the evaluated models and phylogenetic trees were gathered to serve as filter options in the database. These filter options allow the user to find alignments, models and/or trees that fit the data they want to simulate.

So far, EvoNAPS holds over 22,600 DNA and over 6,600 protein alignments that were gathered from three published sources. The sources are the online repository of Rob Lanfear (Lanfear, 2019), the PANDIT database (Whelan et al., 2006) and the OrthoMaM database (Douzery et al., 2014). The EvoNAPS database holds over 64,000 phylogenetic trees (ML or *initial*) with the corresponding branch length estimates. The database provides model parameter estimates of 286 different DNA models and 364 different protein models. Overall, the parameter estimates of over 6.3 million model evaluations are stored in EvoNAPS.

Additional alignments will be added to the database to make the data in EvoNAPS more representative. This expansion can be easily achieved by using the custom-made workflow that was discussed in chapter 2.2. Potential future sources for additional alignments are the Selectome (Moretti et al., 2014) and TreeBase (Carroll et al., 2007) databases.

Furthermore, a potential future project involves assigning taxonomic identifies according to the NCBI Taxonomy Database (Schoch et al., 2020) to the sequences in the alignments. The identifies can then serve as filters, but also provide a form of measurement for which taxa are well and which are still under-represented in the EvoNAPS database.

Yet another future project could entail including additional models of sequence evolution to the EvoNAPS database. So far, only the restrictive models as discussed in section 1.1 (and listed in A.1 and A.2 in the appendix) were evaluated. Potential additional models that are also implemented in IQ-Tree2 are the Lie-Markov models (Woodhams et al., 2015; Hannaford et al., 2020).

Overall, the EvoNAPS database presents a valuable resource to those studying model-based phylogenetics. EvoNAPS will greatly aid future phylogenetic research, especially when expanded.

Appendix A

Substitution rate matrices

Table A.1: Named DNA substitution rate matrices implemented in IQ-Tree 2.

Model	df	Comment
JC	0	Equal substitution rates and equal base frequencies (Jukes and Cantor, 1969).
K2P	1	Unequal transition/transversion rates and equal base freq (Kimura, 1980).
K3P	2	Three substitution types model and equal base freq (Kimura, 1981).
TNe	2	Like TN but equal base freq.
TPM2	2	AC=AT, AG=CT, CG=GT and equal base freq.
TPM3	2	AC=CG, AG=CT, AT=GT and equal base freq.
F81	3	Equal rates but unequal base freq (Felsenstein, 1981).
TIM2e	3	Like TIM2, see below, but equal base freq.
TIM3e	3	Like TIM3, see below, but equal base freq.
TIMe	3	Like TIM, see below, but equal base freq.
HKY	4	Unequal transition/transversion rates and unequal base freq (Hasegawa et al., 1985).

Cont'd on following page

Table A.1, cont'd

Model	**df**	**Explanation**
TVMe	4	Like TVM but equal base freq.
K3Pu	5	Like K3P but unequal base freq.
SYM	5	Symmetric model with unequal rates but equal base freq (Zharkikh, 1994).
TN	5	Like HKY but unequal purine/pyrimidine rates (Tamura and Nei, 1993).
TPM2u	5	Like TPM2 but unequal base freq.
TPM3u	5	Like TPM3 but unequal base freq.
TIM	6	Transition model, AC=GT, AT=CG and unequal base freq.
TIM2	6	AC=AT, CG=GT and unequal base freq.
TIM3	6	AC=CG, AT=GT and unequal base freq.
TVM	7	Transversion model, AG=CT and unequal base freq.
GTR	8	General time reversible model with unequal rates and unequal base freq (Tavaré, 1986).

Table A.2: Named protein substitution rate matrices implemented in IQ-Tree 2.

Model	Region	Explanation
Blosum62	nuclear	BLOcks SUbstitution Matrix (Henikoff and Henikoff, 1992).
cpREV	chloroplast	chloroplast matrix (Adachi and Hasegawa, 1996).
Dayhoff	nuclear	General matrix.
DCMut	nuclear	Revised Dayhoff matrix (Kosiol and Goldman, 2005).
FLAVI	viral	Flavivirus (Le and Vinh, 2020)
FLU	viral	Influenza virus (Dang et al., 2010).
HIVb	viral	HIV between-patient matrix HIV-Bm (Nickle et al., 2007).
HIVw	viral	HIV within-patient matrix HIV-Wm (Nickle et al., 2007).
JTT	nuclear	General matrix (Jones et al., 1992).
JTTDCMut	nuclear	Revised JTT matrix (Kosiol and Goldman, 2005).
LG	nuclear	General matrix (Le and Gascuel, 2008).
mtART	mitochondrial	Mitochondrial Arthropoda (Abascal et al., 2007).
mtMAM	mitochondrial	Mitochondrial Mammalia (Yang et al., 1998).
mtREV	mitochondrial	Mitochondrial Vertebrate (Adachi and Hasegawa, 1996).
mtZOA	mitochondrial	Mitochondrial Metazoa (Rota-Stabelli et al., 2009).
mtMet	mitochondrial	Mitochondrial Metazoa (Le et al., 2017).
mtVer	mitochondrial	Mitochondrial Vertebrate (Le et al., 2017).
mtIn	mitochondrial	Mitochondrial Inverterbrate (Le et al., 2017).
PMB	nuclear	Probability Matrix from Blocks, revised BLOSUM matrix (Veerassamy et al., 2003).
Q.bird	nuclear	Q matrix estimated for birds (Minh et al., 2021).

Cont'd on following page

Table A.2, cont'd

Model	**Region**	**Explanation**
Q.insect	nuclear	Q matrix estimated for insects (Minh et al., 2021).
Q.mammal	nuclear	Q matrix estimated for mammals (Minh et al., 2021).
Q.pfam	nuclear	General matrix estimated from Pfam version 31 (2017) database.
Q.plant	nuclear	Q matrix estimated for plants (Minh et al., 2021).
Q.yeast	nuclear	Q matrix estimated for insects (Minh et al., 2021).
rtREV	viral	Retrovirus (Dimmic et al., 2002).
VT	nuclear	General 'Variable Time' matrix (Müller and Vingron, 2000).
WAG	nuclear	General matrix (Whelan et al., 2006).

Appendix B

Tables of the EvoNAPS database

A detailed description of the tables found in the EvoNAPS database can be found in the sub-chapters below.

Note, that sometimes the tables containing information regarding DNA and protein alignments are identical (e.g., the *alignments* tables). However, in some tables there are differences in the number and kind of columns (e.g., *sequences*, *modelparameters*, *trees* tables).

B.1 Table: *dataorigin*

Content: The *dataorigin* table holds information regarding the original sources of the alignments in the EvoNAPS database.

Constraints:

- **PRIMARY KEY** (DATABASE_KEY)
- **UNIQUE KEY** (DATABASE_ID)

Table B.1: The *dataorigin* table.

Column Name	Datatype	Comment
DATABASE_KEY	int(11)	Autoincremented primary key.
DATABASE_ID	varchar(100)	This field holds the name of the source database, which in turn serves as the ID of said database. The entries of this column must be unique.
DOI	varchar(100)	States the DOI of the paper describing the source database, should there exist one.
PUBMED_ID	varchar(100)	States the PUBMED-ID of the paper describing the source database, should there exist one.
LAST_UPDATED	varchar(100)	States the date the source database was last updated, if available.
SEQ_TYPE	varchar(100)	States whether the source database holds of DNA and/or protein alignments.
DESCRIPTION	text	A text field that gives a short description of the source database.
SIZE	text	States the number of alignments the source database holds.
COMMENT	text	An optional text field for any comments regarding the source database.

B.2 Table: *aa_models*

Content: The *aa_models* table lists the different protein substitution rate matrices that were tested in the EvoNAPS workflow and includes the assumed amino acid frequencies and substitution rates.

Constraints:

- **PRIMARY KEY** (MODEL_KEY)
- **UNIQUE KEY** (MODEL_NAME)

Table B.2: The *aa_models* table.

Column Name	**Datatype**	**Comment**
MODEL_KEY	int(11)	Autoincremented primary key.
MODEL_NAME	varchar(100)	Name of the protein model (substitution rate matrix). The name must be unique.
REGION	varchar(50)	States the region of the cell where the proteins from which the substitution rate matrix was derived from are abundant. Optional, default is NULL.
EXPLANATION	varchar(100)	This field contains a short description of the model.
STAT_DIS_TYPE	varchar(50)	This field states whether the state frequencies of the stationary distribution assumed in the model are empirical (counted frequencies from the alignment) or if they are predefined by the model.
FREQ_A	decimal(10,9)	Either NULL if STAT_DIS_TYPE= 'empirical'. Else, the frequency of the amino acid alanine (A) assumed by the model.
...	...	...
FREQ_V	decimal(10,9)	Either NULL if STAT_DIS_TYPE= 'empirical'. Else, the frequency of the amino acid tyrosine (Y) assumed by the model.
RATE_AR	decimal(15,9)	The substitution rate from aa A to aa R assumed by the model.
...	...	...
RATE_YV	decimal(15,9)	The substitution rate from aa Y to aa V assumed by the model.

B.3 Table: *dna_models*

Content: The *dna_models* table lists the different DNA substitution rate matrices that were tested in the EvoNAPS workflow.

Constraints:

- **PRIMARY KEY** (MODEL_KEY)
- **UNIQUE KEY** (MODEL_NAME)

Table B.3: The *dna_models* table.

Column Name	**Datatype**	**Comment**
MODEL_KEY	int(11)	Autoincremented primary key.
MODEL_NAME	varchar(100)	Name of the DNA model (substitution rate matrix). The name must be unique.
FREE_PARAMETERS	int(11)	States the number of free parameters of the model.
BASE_FREQUENCIES	varchar(30)	States whether the assumed base frequencies of the model are uniform (0.25 for each base) or unequal.
SUBSTITUTION_RATES	varchar(100)	States (possible) restrictions the model has on the substitution rates.
EXPLANATION	varchar(100)	This field contains a short description of the model.
SUBSTITUTION_CODE	varchar(100)	This field shows the substitution code of the rate matrix.

B.4 Table: *aa_alignments*

Content: The *aa_alignments* table holds general information and characteristics regarding each protein alignment in the database.

Constraints:

- **PRIMARY KEY** (ALI_KEY)
- **UNIQUE KEY** (ALI_ID)
- **FOREIGN KEY** (FROM_DATABASE) REFERENCES *dataorigin* (DATABASE_ID)

Table B.4: The *aa_alignments* table.

Column Name	Datatype	Comment
ALI_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID). Must be unique.
FROM_ DATABASE	varchar(100)	States from which original database the alignemnt stems from (e.g. PANDIT). Serves as foreign key to connect to the dataorigin table.
DESCRIPTION	varchar(100)	A field that can hold an optional comment regarding the alignment. This can be left blank and the default value is accordingly NULL.
SEQUENCES	int(11)	This column states how many seqeunces (taxa) the alignemnt holds.
COLUMNS	int(11)	This column states how many sites (columns) the alignemnt has / states the length of the alignment.
PARSIMONY_ INFORMATIVE_ SITES	int(11)	States the number of parsimony informative sites in alignment.
SINGELTON_ SITES	int(11)	States the number of singelton sites in alignment.

Cont'd on following page

Table B.4, cont'd

Column Name	Datatype	Comment
CONSTANT_SITES	int(11)	States the number of constant sites in alignment.
FRAC_WILDCARDS_GAPS	decimal(5,4)	States the fraction of wildcards and gaps in the alignment.
DISTINCT_PATTERNS	int(11)	States the number of distinct patterns in alignment.
FAILED_CHI2	int(11)	States the number of sequences that failed the chi2 (chi-squared) test. The test examines whether the nucleotide composition of the sequences matches the mean nucleotide frequencies across all sequences.
IDENTICAL_SEQ	int(11)	States the number of identical sequences in the alignment, should there be any. Default is NULL.
EXCLUDED_SEQ	int(11)	States the number of excluded sequences in the alignment, should there be any. Default is NULL.

B.5 Table: *dna_alignments*

Content: The *dna_alignments* table lists the different DNA substitution rate matrices that were tested in the EvoNAPS workflow.

Constraints:

- **PRIMARY KEY** (ALI_KEY)
- **UNIQUE KEY** (ALI_ID)
- **FOREIGN KEY** (FROM_DATABASE) REFERENCES *dataorigin* (DATABASE_ID)

Table B.5: The *dna_alignments* table.

Column Name	Datatype	Comment
ALI_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID). Must be unique.
FROM_ DATABASE	varchar(100)	States from which original database the alignemnt stems from (e.g. PANDIT). Serves as foreign key to connect to the dataorigin table.
DESCRIPTION	varchar(100)	A field that can hold an optional comment regarding the alignment. This can be left blank and the default value is accordingly NULL.
SEQUENCES	int(11)	This column states how many seqeunces (taxa) the alignemnt holds.
COLUMNS	int(11)	This column states how many sites (columns) the alignemnt has / states the length of the alignment.
PARSIMONY_ INFORMATIVE_ SITES	int(11)	States the number of parsimony informative sites in alignment.
SINGELTON_ SITES	int(11)	States the number of singelton sites in alignment.
CONSTANT_SITES	int(11)	States the number of constant sites in alignment.
FRAC_ WILDCARDS_GAPS	decimal(5,4)	States the fraction of wildcards and gaps in the alignment.
DISTINCT_ PATTERNS	int(11)	States the number of distinct patterns in alignment.
FAILED_CHI2	int(11)	States the number of sequences that failed the chi2 (chi-squared) test. The test examines whether the nucleotide composition of the sequences matches the mean nucleotide frequencies across all sequences.

Cont'd on following page

Table B.5, cont'd

Column Name	Datatype	Comment
IDENTICAL_SEQ	int(11)	States the number of identical sequences in the alignment, should there be any. Default is NULL.
EXCLUDED_SEQ	int(11)	States the number of excluded sequences in the alignment, should there be any. Default is NULL.

B.6 Table: *aa_sequences*

Content: The *aa_sequences* table holds the sequences of each protein alignment in the EvoNAPS database as well as information regarding each sequence.

Constraints:

- **PRIMARY KEY** (SEQ_KEY)
- **UNIQUE KEY** (ALI_ID,SEQ_INDEX)
- **FOREIGN KEY** (ALI_ID) REFERENCES *aa_alignments* (ALI_ID)

Table B.6: The *aa_sequences* table.

Column Name	Datatype	Comment
SEQ_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID). Serves as foreign key to connect to the aa_alignments table.
SEQ_INDEX	int(11)	This column holds the unique index (integer starting with 1) for each sequence of an alignment.
SEQ_NAME	varchar(250)	States the name of the sequence as it appears in the original alignment.
FRAC_WILDCARDS_GAPS	decimal(10,9)	States the fraction of wildcards and gaps in the sequence.

Cont'd on following page

Table B.6, cont'd

Column Name	Datatype	Comment
CHI2_P_VALUE	decimal(7,2)	States the p-value of the Chi-Square test for the sequence. The Chi-Square test tests whether the amino acid composition of the sequence fits the mean aa frequencies across all sequences in the alignment.
CHI2_PASSED	tinyint(1)	States whether the sequence passed (1) or failed (0) the Chi-Square test.
EXCLUDED	int(11)	States whether the sequence has been excluded from IQ-Tree calculations (without the flag `--keep-ident`). IQ-Tree excludes a sequence from its computations if there already exist at least two identical sequences in the alignment.
IDENTICAL_TO	varchar(10000)	States to which sequence(s) the sequence is identical to, if such (a) sequence(s) exist(s).
FREQ_A	decimal(10,9)	The frequency of the amino acid alanine (A) in the sequence.
...		...
FREQ_V	decimal(10,9)	The frequency of the amino acid tyrosine (V) in the sequence.
SEQ	mediumtext	This text field contains the sequence (with wildcards and gaps) as it appears in the alignment.

B.7 Table: *dna_sequences*

Content: The *dna_sequences* table holds the sequences of each DNA alignment in the EvoNAPS database as well as information regarding each sequence.

Constraints:

- **PRIMARY KEY** (SEQ_KEY)
- **UNIQUE KEY** (ALI_ID,SEQ_INDEX)
- **FOREIGN KEY** (ALI_ID) REFERENCES *dna_alignments* (ALI_ID)

Table B.7: The *dna_sequences* table.

Column Name	**Datatype**	**Comment**
SEQ_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID). Serves as foreign key to connect to the dna_alignments table.
SEQ_INDEX	int(11)	This column holds the unique index (integer starting with 1) for each sequence of an alignment.
SEQ_NAME	varchar(250)	States the name of the sequence as it appears in the original alignment.
FRAC_ WILDCARDS_GAPS	decimal(10,9)	States the fraction of wildcards and gaps in the sequence.
CHI2_P_VALUE	decimal(7,2)	States the p-value of the Chi-Square test for the sequence. The Chi-Square test tests whether the nucleotide composition of the sequence fits the mean dna frequencies across all sequences in the alignment.
CHI2_PASSED	tinyint(1)	States whether the sequence passed (1) or failed (0) the Chi-Square test.
EXCLUDED	int(11)	States whether the sequence has been excluded from IQ-Tree calculations (without the flag `--keep-ident`). IQ-Tree excludes a sequence from its computations if there already exist at least two identical sequences in the alignment.
IDENTICAL_TO	varchar(10000)	States to which sequence(s) the sequence is identical to, if such (a) sequence(s) exist(s).
FREQ_A	decimal(10,9)	The frequency of the base adenine (A) in the sequence.
FREQ_C	decimal(10,9)	The frequency of the base cytosine (C) in the sequence.
FREQ_G	decimal(10,9)	The frequency of the base guanine (G) in the sequence.
FREQ_T	decimal(10,9)	The frequency of the base thymine (T) in the sequence.

Cont'd on following page

Table B.7, cont'd

Column Name	Datatype	Comment
SEQ	mediumtext	This text field contains the sequence (with wildcards and gaps) as it appears in the alignment.

B.8 Table: *aa_modelparameters*

Content: The *aa_modelparameters* table holds the results of model selection. The performance of each evaluated model (LogL, AIC, BIC,...) is documented as well as the parameters of the model (state frequencies, rates, shape parameter alpha,...).

Constraints:

- **PRIMARY KEY** (MODELTEST_KEY)
- **KEY** (BASE_MODEL)
- **UNIQUE KEY** (ALI_ID,TIME_STAMP,MODEL)
- **FOREIGN KEY** (ALI_ID) REFERENCES *aa_alignments* (ALI_ID)
- **FOREIGN KEY** (BASE_MODEL) REFERENCES *aa_models* (MODEL_NAME)

Table B.8: The *aa_modelparameters* table.

Column Name	Datatype	Comment
MODELTEST_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID).
IQTREE_VERSION	varchar(100)	
RANDOM_SEED	int(11)	The random number seed used by IQ-Tree.
TIME_STAMP	datetime	The timestamp as it appears in the *.iqtree output file. The timestamp enables mapping of the tested model to one IQ-Tree run.

Cont'd on following page

Table B.8, cont'd

Column Name	**Datatype**	**Comment**
MODEL_TYPE	varchar(100)	The type of model testing or the type of models that were tested in the IQ-Tree run. Will mostly be MF (the models included in the default ModelFinder algorithm).
KEEP_IDENT	tinyint(1)	Boolean stating whether the `--keep-ident` flag has been enabled (1) or disabled (0) in the IQ-Tree run.
MODEL	varchar(100)	Name of the tested model
BASE_MODEL	varchar(100)	Name of the substitution rate matrix used in the model.
MODEL_RATE_HETEROGENEITY	varchar(100)	Name of the model of rate heterogeneity (should one have been employed).
NUM_RATE_CAT	int(11)	Number of rate categories assumed by the model.
LOGL	decimal(21,9)	Logarithmic likelihood
AIC	decimal(21,9)	
WEIGHTED_AIC	float	
CONFIDENCE_AIC	tinyint(1)	Boolean stating whether the weighted AIC is above 0.05 (1) or under (0).
AICC	decimal(21,9)	
WEIGHTED_AICC	float	
CONFIDENCE_AICC	tinyint(1)	Boolean stating whether the weighted AICC is above 0.5 (1) or under (0).
BIC	decimal(21,9)	
WEIGHTED_BIC	float	
CONFIDENCE_BIC	tinyint(1)	Boolean stating whether the weighted BIC is above 0.05 (1) or under (0).
CAIC	decimal(21,9)	
WEIGHTED_CAIC	float	
CONFIDENCE_CAIC	tinyint(1)	Boolean stating whether the weighted CAIC is above 0.05 (1) or under (0).
ABIC	decimal(21,9)	
WEIGHTED_ABIC	float	

Cont'd on following page

Table B.8, cont'd

Column Name	Datatype	Comment
CONFIDENCE_ABIC	tinyint(1)	Boolean stating whether the weighted ABIC is above 0.05 (1) or under (0).
NUM_FREE_PARAMETERS	int(11)	Number of free parameters (=NUM_MODEL_PARAMETERS+NUM_BRANCHES).
NUM_MODEL_PARAMETERS	int(11)	Number of free parameters of the model of sequence evolution
NUM_BRANCHES	int(11)	Number of branches in the phylogenetic tree. In a fully resolved tree: 2n-3 (with n taxa).
TREE_LENGTH	decimal(15,9)	Length of the tree (might differ for the different models as the branch lengths are being re-estimated during model evaluation).
PROP_INVAR	decimal(10,9)	Proportion of invariable sites in case the +I model of rate heterogeneity was employed. Else, NULL.
ALPHA	decimal(15,9)	Shape parameter alpha should an Gamma +G4 model have been employed. Else, NULL.
STAT_FREQ_TYPE	varchar(100)	This field states whether the state frequencies of the stationary distribution assumed in the model are empirical (counted frequencies from the alignment) or if they are predefined by the model (model).
STAT_FREQ_A	decimal(10,9)	The stationary frequency of the amino acid alanine (A) assumed by the model.
...		...
STAT_FREQ_V	decimal(10,9)	The stationary frequency of the amino acid tyrosine (V) assumed by the model.
PROP_CAT_1	decimal(10,9)	The proportion of the first rate category (should the model assume different rates across sites).

Cont'd on following page

Table B.8, cont'd

Column Name	Datatype	Comment
REL_RATE_CAT_1	decimal(15,9)	The rate of the first rate category (should the model assume different rates across sites).
...		...
PROP_CAT_10	decimal(10,9)	The proportion of the tenth rate category (should there exist one).
REL_RATE_CAT_10	decimal(15,9)	The rate of the tenth rate category (should there exist one).

B.9 Table: *dna_modelparameters*

Content: The *dna_modelparameters* table holds the results of model selection. The performance of each evaluated model (LogL, AIC, BIC,...) is documented as well as the parameters of the model (state frequencies, rates, shape parameter alpha,...).

Constraints:

- **PRIMARY KEY** (MODELTEST_KEY)
- **KEY** (BASE_MODEL)
- **UNIQUE KEY** (ALI_ID,TIME_STAMP,MODEL)
- **FOREIGN KEY** (ALI_ID) REFERENCES *dna_alignments* (ALI_ID)
- **FOREIGN KEY** (BASE_MODEL) REFERENCES *dna_models* (MODEL_NAME)

Table B.9: The *dna_modelparameters* table.

Column Name	Datatype	Comment
MODELTEST_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID).
IQTREE_VERSION	varchar(100)	
RANDOM_SEED	int(11)	The random number seed used by IQ-Tree.

Cont'd on following page

Table B.9, cont'd

Column Name	Datatype	Comment
TIME_STAMP	datetime	The timestamp as it appears in the *.iqtree output file. The timestamp enables mapping of the tested model to one IQ-Tree run.
MODEL_TYPE	varchar(100)	The type of model testing or the type of models that were tested in the IQ-Tree run. Will mostly be MF (the models included in the default ModelFinder algorithm).
KEEP_IDENT	tinyint(1)	Boolean stating whether the `--keep-ident` flag has been enabled (1) or disabled (0) in the IQ-Tree run.
MODEL	varchar(100)	Name of the tested model
BASE_MODEL	varchar(100)	Name of the substitution rate matrix used in the model.
MODEL_RATE_HETEROGENEITY	varchar(100)	Name of the model of rate heterogeneity (should one have been employed).
NUM_RATE_CAT	int(11)	Number of rate categories assumed by the model.
LOGL	decimal(21,9)	Logarithmic likelihood
AIC	decimal(21,9)	
WEIGHTED_AIC	float	
CONFIDENCE_AIC	tinyint(1)	Boolean stating whether the weighted AIC is above 0.05 (1) or under (0).
AICC	decimal(21,9)	
WEIGHTED_AICC	float	
CONFIDENCE_AICC	tinyint(1)	Boolean stating whether the weighted AICC is above 0.5 (1) or under (0).
BIC	decimal(21,9)	
WEIGHTED_BIC	float	
CONFIDENCE_BIC	tinyint(1)	Boolean stating whether the weighted BIC is above 0.05 (1) or under (0).
CAIC	decimal(21,9)	
WEIGHTED_CAIC	float	
CONFIDENCE_CAIC	tinyint(1)	Boolean stating whether the weighted CAIC is above 0.05 (1) or under (0).

Cont'd on following page

Table B.9, cont'd

Column Name	**Datatype**	**Comment**
ABIC	decimal(21,9)	
WEIGHTED_ABIC	float	
CONFIDENCE_ABIC	tinyint(1)	Boolean stating whether the weighted ABIC is above 0.05 (1) or under (0).
NUM_FREE_PARAMETERS	int(11)	Number of free parameters (=NUM_MODEL_PARAMETERS+ NUM_BRANCHES).
NUM_MODEL_PARAMETERS	int(11)	Number of free parameters of the model of sequence evolution
NUM_BRANCHES	int(11)	Number of branches in the phylogenetic tree. In a fully resolved tree: 2n-3 (with n taxa).
TREE_LENGTH	decimal(15,9)	Length of the tree (might differ for the different models as the branch lengths are being re-estimated during model evaluation).
PROP_INVAR	decimal(10,9)	Proportion of invariable sites in case the +I model of rate heterogeneity was employed. Else, NULL.
ALPHA	decimal(15,9)	Shape parameter alpha should an Gamma +G4 model have been employed. Else, NULL.
STAT_FREQ_TYPE	varchar(100)	This field states whether the state frequencies of the stationary distribution assumed in the model are empirical (counted frequencies from the alignment) or if they are predefined by the model (model).
STAT_FREQ_A	decimal(10,9)	The stationary frequency of the base adenine (A) assumed by the model.
STAT_FREQ_C	decimal(10,9)	The stationary frequency of the base guanine (G) assumed by the model.
STAT_FREQ_G	decimal(10,9)	The stationary frequency of the base cytosine (C) assumed by the model.
STAT_FREQ_T	decimal(10,9)	The stationary frequency of the base thymine (T) assumed by the model.

Cont'd on following page

Table B.9, cont'd

Column Name	**Datatype**	**Comment**
RATE_AC	decimal(15,9)	Assumed relative substitution rate from A to C.
RATE_CA	decimal(15,9)	Assumed relative substitution rate from C to A.
...		...
RATE_GT	decimal(15,9)	Assumed relative substitution rate from G to T.
RATE_TG	decimal(15,9)	Assumed relative substitution rate from T to G.
PROP_CAT_1	decimal(10,9)	The proportion of the first rate category (should the model assume different rates across sites).
REL_RATE_CAT_1	decimal(15,9)	The rate of the first rate category (should the model assume different rates across sites).
...		...
PROP_CAT_10	decimal(10,9)	The proportion of the tenth rate category (should there exist one).
REL_RATE_CAT_10	decimal(15,9)	The rate of the tenth rate category (should there exist one).

B.10 Table: *aa_trees*

Content: The *aa_trees* table contains a set of phylogenetic trees as well as the parameters of the assumed model of sequence evolution. The trees are either a fast-ML tree used in the model evaluation or a maximum likelihood (ML) tree inferred using the best-fit model.

Constraints:

- **PRIMARY KEY** (TREE_KEY)
- **KEY** (BASE_MODEL)
- **UNIQUE KEY** (ALI_ID,TIME_STAMP,TREE_TYPE)
- **FOREIGN KEY** (ALI_ID) REFERENCES *aa_alignments* (ALI_ID)
- **FOREIGN KEY** (BASE_MODEL) REFERENCES *aa_models* (MODEL_NAME)

Table B.10: The *aa_trees* table.

Column Name	Datatype	Comment
TREE_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID).
IQTREE_VERSION	varchar(100)	
RANDOM_SEED	int(11)	The random number seed used by IQ-Tree.
TIME_STAMP	datetime	The timestamp as it appears in the *.iqtree output file. The timestamp enables mapping of the tested model to one IQ-Tree run.
MODEL_TYPE	varchar(100)	The type of model testing or the type of models that were tested in the IQ-Tree run. Will mostly be MF (the models included in the default ModelFinder algorithm).
TREE_TYPE	varchar(100)	States the type of the tree. It is either initial (fast ML tree used for model evaluation) or ML (maximum likelihood tree).

Cont'd on following page

Table B.10, cont'd

Column Name	**Datatype**	**Comment**
CHOICE_CRITERIUM	varchar(100)	States the choice criterium used to select the model for the ML tree search. In case of an initial tree, this field is left empty (NULL).
KEEP_IDENT	tinyint(1)	Boolean stating whether the `--keep-ident` flag has been enabled (1) or disabled (0) in the IQ-Tree run.
MODEL	varchar(100)	Name of the tested model
BASE_MODEL	varchar(100)	Name of the substitution rate matrix used in the model.
MODEL_RATE_HETEROGENEITY	varchar(100)	Name of the model of rate heterogeneity (should one have been employed).
NUM_RATE_CAT	int(11)	Number of rate categories assumed by the model.
LOGL	decimal(21,9)	Logarithmic likelihood
UNCONSTRAINED_LOGL	decimal(21,9)	Unconstrained logarithmic likelihood
AIC	decimal(21,9)	
AICC	decimal(21,9)	
BIC	decimal(21,9)	
CAIC	decimal(21,9)	
ABIC	decimal(21,9)	
NUM_FREE_PARAMETERS	int(11)	Number of free parameters (=NUM_MODEL_PARAMETERS+ NUM_BRANCHES).
NUM_MODEL_PARAMETERS	int(11)	Number of free parameters of the model of sequence evolution
NUM_BRANCHES	int(11)	Number of branches in the phylogenetic tree. In a fully resolved tree: 2n-3 (with n taxa).
PROP_INVAR	decimal(10,9)	Proportion of invariable sites in case the +I model of rate heterogeneity was employed. Else, NULL.

Cont'd on following page

Table B.10, cont'd

Column Name	**Datatype**	**Comment**
ALPHA	decimal(15,9)	Shape parameter alpha should an Gamma +G4 model have been employed. Else, NULL.
STAT_FREQ_TYPE	varchar(100)	This field states whether the state frequencies of the stationary distribution assumed in the model are empirical (counted frequencies from the alignment) or if they are predefined by the model (model).
STAT_FREQ_A	decimal(10,9)	The frequency of the amino acid alanine (A) assumed by the model.
...		...
STAT_FREQ_V	decimal(10,9)	The frequency of the amino acid tyrosine (V) assumed by the model.
PROP_CAT_1	decimal(10,9)	The proportion of the first rate category (should the model assume different rates across sites).
REL_RATE_CAT_1	decimal(15,9)	The rate of the first rate category (should the model assume different rates across sites).
...		...
PROP_CAT_10	decimal(10,9)	The proportion of the tenth rate category (should there exist one).
REL_RATE_CAT_10	decimal(15,9)	The rate of the tenth rate category (should there exist one).
TREE_LENGTH	decimal(15,9)	Total length of the tree (sum of all branch lengths).
SUM_IBL	decimal(15,9)	Sum of internal branch lengths
TREE_DIAMETER	decimal(15,9)	The tree diameter states the furthest distance (sum of BLs) between two taxa in the tree.
DIST_MIN	decimal(15,9)	Minimal distance between two sequences in the alignment caculated using the best-fit model. In case of initial tree, this field will be NULL.

Cont'd on following page

Table B.10, cont'd

Column Name	**Datatype**	**Comment**
DIST_MAX	decimal(15,9)	Maximum distance between two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
DIST_MEAN	decimal(15,9)	Mean distance between two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
DIST_MEDIAN	decimal(15,9)	Median distance between two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
DIST_VAR	decimal(15,9)	Variation in distances between any two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
BL_MIN	decimal(15,9)	Shortest branch in the tree
BL_MAX	decimal(15,9)	Longest branch in the tree.
BL_MEAN	decimal(15,9)	Mean branch length in the tree.
BL_MEDIAN	decimal(15,9)	Median branch length in the tree.
BL_VAR	decimal(15,9)	Variation in branch lengths in the tree.
IBL_MIN	decimal(15,9)	Shortest internal branch in the tree
IBL_MAX	decimal(15,9)	Longest internal branch in the tree.
IBL_MEAN	decimal(15,9)	Mean internal branch length in the tree.
IBL_MEDIAN	decimal(15,9)	Median internal branch length in the tree.
IBL_VAR	decimal(15,9)	Variation in internal branch lengths in the tree.
EBL_MIN	decimal(15,9)	Shortest external branch in the tree
EBL_MAX	decimal(15,9)	Longest external branch in the tree.
EBL_MEAN	decimal(15,9)	Mean external branch length in the tree.
EBL_MEDIAN	decimal(15,9)	Median external branch length in the tree.
EBL_VAR	decimal(15,9)	Variation in external branch lengths in the tree.

Cont'd on following page

Table B.10, cont'd

Column Name	Datatype	Comment
POT_LBA_7	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 7 times larger than the short and internal branch.
POT_LBA_8	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 8 times larger than the short and internal branch.
POT_LBA_9	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 9 times larger than the short and internal branch.
POT_LBA_10	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 10 times larger than the short and internal branch.
NEWICK_STRING	mediumtext	This field contains the Newick string of the phylogenetic tree.

B.11 Table: *dna_trees*

Content: The *dna_trees* table contains a set of phylogenetic trees as well as the parameters of the assumed model of sequence evolution. The trees are either a fast-ML tree used in the model evaluation or a maximum likelihood (ML) tree inferred using the best-fit model.

Constraints:

- **PRIMARY KEY** (TREE_KEY)
- **KEY** (BASE_MODEL)

- **UNIQUE KEY** (ALI_ID,TIME_STAMP,TREE_TYPE)
- **FOREIGN KEY** (ALI_ID) REFERENCES *dna_alignments* (ALI_ID)
- **FOREIGN KEY** (BASE_MODEL) REFERENCES *dna_models* (MODEL_NAME)

Table B.11: The *dna_trees* table.

Column Name	**Datatype**	**Comment**
TREE_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID).
IQTREE_VERSION	varchar(100)	
RANDOM_SEED	int(11)	The random number seed used by IQ-Tree.
TIME_STAMP	datetime	The timestamp as it appears in the .iqtree output file. The timestamp enables mapping of the tested model to one IQ-Tree run.
MODEL_TYPE	varchar(100)	The type of model testing or the type of models that were tested in the IQ-Tree run. Will mostly be MF (the models included in the default ModelFinder algorithm).
TREE_TYPE	varchar(100)	States the type of the tree. It is either initial (fast ML tree used for model evaluation) or ML (maximum likelihood tree).
CHOICE_CRITERIUM	varchar(100)	States the choice criterium used to select the model for the ML tree search. In case of an initial tree, this field is left empty (NULL).
KEEP_IDENT	tinyint(1)	Boolean stating whether the `--keep-ident` flag has been enabled (1) or disabled (0) in the IQ-Tree run.
MODEL	varchar(100)	Name of the tested model
BASE_MODEL	varchar(100)	Name of the substitution rate matrix used in the model.
MODEL_RATE_HETEROGENEITY	varchar(100)	Name of the model of rate heterogeneity (should one have been employed).
NUM_RATE_CAT	int(11)	Number of rate categories assumed by the model.

Cont'd on following page

Table B.11, cont'd

Column Name	**Datatype**	**Comment**
LOGL	decimal(21,9)	Logarithmic likelihood
UNCONSTRAINED_LOGL	decimal(21,9)	Unconstrained logarithmic likelihood
AIC	decimal(21,9)	
AICC	decimal(21,9)	
BIC	decimal(21,9)	
CAIC	decimal(21,9)	
ABIC	decimal(21,9)	
NUM_FREE_PARAMETERS	int(11)	Number of free parameters (=NUM_MODEL_PARAMETERS+ NUM_BRANCHES).
NUM_MODEL_PARAMETERS	int(11)	Number of free parameters of the model of sequence evolution
NUM_BRANCHES	int(11)	Number of branches in the phylogenetic tree. In a fully resolved tree: 2n-3 (with n taxa).
PROP_INVAR	decimal(10,9)	Proportion of invariable sites in case the +I model of rate heterogeneity was employed. Else, NULL.
ALPHA	decimal(15,9)	Shape parameter alpha should an Gamma +G4 model have been employed. Else, NULL.
STAT_FREQ_TYPE	varchar(100)	This field states whether the state frequencies of the stationary distribution assumed in the model are empirical (counted frequencies from the alignment) or if they are predefined by the model (model).
STAT_FREQ_A	decimal(10,9)	The frequency of the base adenine (A) assumed by the model.
STAT_FREQ_C	decimal(10,9)	The frequency of the base guanine (G) assumed by the model.
STAT_FREQ_G	decimal(10,9)	The frequency of the base cytosine (C) assumed by the model.

Cont'd on following page

Table B.11, cont'd

Column Name	**Datatype**	**Comment**
STAT_FREQ_T	decimal(10,9)	The frequency of the base thymine (T) assumed by the model.
RATE_AC	decimal(15,9)	Assumed relative substitution rate from A to C.
RATE_CA	decimal(15,9)	Assumed relative substitution rate from C to A.
...		...
RATE_GT	decimal(15,9)	Assumed relative substitution rate from G to T.
RATE_TG	decimal(15,9)	Assumed relative substitution rate from T to G.
PROP_CAT_1	decimal(10,9)	The proportion of the first rate category (should the model assume different rates across sites).
REL_RATE_CAT_1	decimal(15,9)	The rate of the first rate category (should the model assume different rates across sites).
...		...
PROP_CAT_10	decimal(10,9)	The proportion of the tenth rate category (should there exist one).
REL_RATE_CAT_10	decimal(15,9)	The rate of the tenth rate category (should there exist one).
TREE_LENGTH	decimal(15,9)	Total length of the tree (sum of all branch lengths).
SUM_IBL	decimal(15,9)	Sum of internal branch lengths
TREE_DIAMETER	decimal(15,9)	The tree diameter states the furthest distance (sum of BLs) between two taxa in the tree.
DIST_MIN	decimal(15,9)	Minimal distance between two sequences in the alignment caculated using the best-fit model. In case of initial tree, this field will be NULL.

Cont'd on following page

Table B.11, cont'd

Column Name	**Datatype**	**Comment**
DIST_MAX	decimal(15,9)	Maximum distance between two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
DIST_MEAN	decimal(15,9)	Mean distance between two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
DIST_MEDIAN	decimal(15,9)	Meadian distance between two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
DIST_VAR	decimal(15,9)	Variation in distances between any two sequences in the alignment calculated using the best-fit model. In case of initial tree, this field will be NULL.
BL_MIN	decimal(15,9)	Shortest branch in the tree
BL_MAX	decimal(15,9)	Longest branch in the tree.
BL_MEAN	decimal(15,9)	Mean branch length in the tree.
BL_MEDIAN	decimal(15,9)	Median branch length in the tree.
BL_VAR	decimal(15,9)	Variation in branch lengths in the tree.
IBL_MIN	decimal(15,9)	Shortest internal branch in the tree
IBL_MAX	decimal(15,9)	Longest internal branch in the tree.
IBL_MEAN	decimal(15,9)	Mean internal branch length in the tree.
IBL_MEDIAN	decimal(15,9)	Median internal branch length in the tree.
IBL_VAR	decimal(15,9)	Variation in internal branch lengths in the tree.
EBL_MIN	decimal(15,9)	Shortest external branch in the tree
EBL_MAX	decimal(15,9)	Longest external branch in the tree.
EBL_MEAN	decimal(15,9)	Mean external branch length in the tree.
EBL_MEDIAN	decimal(15,9)	Median external branch length in the tree.
EBL_VAR	decimal(15,9)	Variation in external branch lengths in the tree.

Cont'd on following page

Table B.11, cont'd

Column Name	Datatype	Comment
POT_LBA_7	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 7 times larger than the short and internal branch.
POT_LBA_8	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 8 times larger than the short and internal branch.
POT_LBA_9	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 9 times larger than the short and internal branch.
POT_LBA_10	int(11)	States if there exists a potential long branch attraction (LBA) problem in the tree. Assuming that the long branches need to be at least 10 times larger than the short and internal branch.
NEWICK_STRING	mediumtext	This field contains the Newick string of the phylogenetic tree.

B.12 Table: *aa_branches*

Content: The *aa_branches* table contains information regarding the branches of the phylogenetic trees stored in the *aa_trees* table. Each line contains information regarding one branch such as the branch type, the branch length, the splitsize, etc.

Constraints:

- **PRIMARY KEY** (BRANCH_KEY)
- **UNIQUE KEY** (ALI_ID,BRANCH_INDEX,TIME_STAMP,TREE_TYPE)
- **KEY** (ALI_ID,BRANCH_INDEX,TREE_TYPE)
- **FOREIGN KEY** (ALI_ID,TIME_STAMP,TREE_TYPE) REFERENCES *aa_trees* (ALI_ID,TIME_STAMP,TREE_TYPE)

Table B.12: The *aa_branches* table.

Column Name	Datatype	Comment
BRANCH_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID).
TIME_STAMP	datetime	The timestamp as it appears in the *.iqtree output file. The timestamp, paired with the alignment ID and tree type, enables the mapping of each branch to a phylogenetic tree in the dna_trees table.
TREE_TYPE	varchar(100)	The type of tree: initial or ML. The tree type, paired with the alignment ID and time stamp, enables the mapping of each branch to a phylogenetic tree in the dna_trees table.
BRANCH_INDEX	int(11)	Index of the branch. Should the branch be external, then the index connected to a taxon coincides with the SEQ_INDEX of the corresponding sequence in the aa_sequences table with the same ALI_ID.
BRANCH_TYPE	varchar(30)	States the type of branch, either internal or external
BL	decimal(15,9)	Branch length.

Cont'd on following page

Table B.12, cont'd

Column Name	Datatype	Comment
SPLIT_SIZE	int(11)	States the split size (number of taxa in the smaller subtree). For external branches, the splitsize is always 1.
MIN_PATH_1	decimal(15,9)	Shortest path length to the leaves in the smaller subtree.
MAX_PATH_1	decimal(15,9)	Longest path length to the leaves in the smaller subtree.
MEAN_PATH_1	decimal(15,9)	Mean path length to the leaves in the smaller subtree.
MEDIAN_PATH_1	decimal(15,9)	Median path length to the leaves in the smaller subtree.
MIN_PATH_2	decimal(15,9)	Shortest path length to the leaves in the larger subtree.
MAX_PATH_2	decimal(15,9)	Longest path length to the leaves in the larger subtree.
MEAN_PATH_2	decimal(15,9)	Mean path length to the leaves in the larger subtree.
MEDIAN_PATH_2	decimal(15,9)	Median path length to the leaves in the larger subtree.

B.13 Table: *dna_branches*

Content: The *dna_branches* table contains information regarding the branches of the phylogenetic trees stored in the *dna_trees* table. Each line contains information regarding one branch such as the branch type, the branch length, the splitsize, etc.

Constraints:

- **PRIMARY KEY** (BRANCH_KEY)
- **UNIQUE KEY** (ALI_ID,BRANCH_INDEX,TIME_STAMP,TREE_TYPE)
- **KEY** (ALI_ID,BRANCH_INDEX,TREE_TYPE)
- **FOREIGN KEY** (ALI_ID,TIME_STAMP,TREE_TYPE) REFERENCES *dna_trees* (ALI_ID,TIME_STAMP,TREE_TYPE)

Table B.13: The *dna_branches* table.

Column Name	Datatype	Comment
BRANCH_KEY	int(11)	Autoincremented primary key.
ALI_ID	varchar(250)	Name of the alignment (alignment ID).
TIME_STAMP	datetime	The timestamp as it appears in the *.iqtree output file. The timestamp, paired with the alignment ID and tree type, enables the mapping of each branch to a phylogenetic tree in the dna_trees table.
TREE_TYPE	varchar(100)	The type of tree: initial or ML. The tree type, paired with the alignment ID and time stamp, enables the mapping of each branch to a phylogenetic tree in the dna_trees table.
BRANCH_INDEX	int(11)	Index of the branch. Should the branch be external, then the index connected to a taxon coincides with the SEQ_INDEX of the corresponding sequence in the aa_sequences table with the same ALI_ID.
BRANCH_TYPE	varchar(30)	States the type of branch, either internal or external
BL	decimal(15,9)	Branch length.
SPLIT_SIZE	int(11)	States the split size (number of taxa in the smaller subtree). For external branches, the splitsize is always 1.
MIN_PATH_1	decimal(15,9)	Shortest path length to the leaves in the smaller subtree.
MAX_PATH_1	decimal(15,9)	Longest path length to the leaves in the smaller subtree.
MEAN_PATH_1	decimal(15,9)	Mean path length to the leaves in the smaller subtree.
MEDIAN_PATH_1	decimal(15,9)	Median path length to the leaves in the smaller subtree.
MIN_PATH_2	decimal(15,9)	Shortest path length to the leaves in the larger subtree.
MAX_PATH_2	decimal(15,9)	Longest path length to the leaves in the larger subtree.

Cont'd on following page

Table B.13, cont'd

Column Name	Datatype	Comment
MEAN_PATH_2	decimal(15,9)	Mean path length to the leaves in the larger subtree.
MEDIAN_PATH_2	decimal(15,9)	Median path length to the leaves in the larger subtree.

Bibliography

Abascal, F., Posada, D., and Zardoya, R. (2007). MtArt: A new model of amino acid replacement for arthropoda. *Mol. Biol. Evol.*, 24:1–5.

Adachi, J. and Hasegawa, M. (1996). Model of amino acid substitution in proteins encoded by mitochondrial DNA. *J. Mol. Evol.*, 42:459–468.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Proceedings of the 2nd International Symposium on Information Theory (ISIT 1971)*, pages 267–281, Budapest, Hungary. Akademiai Kiado.

Bozdogan, H. (1987). Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370.

Bryant, D., Galtier, N., and Poursat, M.-A. (2005). Likelihood calculation in molecular phylogenetics. In Gascuel, O., editor, *Mathematics of Evolution and Phylogeny*, pages 33–62. Oxford University Press, Oxford, UK.

Burnham, K. and Anderson, D. (2003). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer New York.

Burnham, K. P. and Anderson, D. R. (2001). Kullback-leibler information as a basis for strong inference in ecological studies. *Wildlife Research*, 28:111–119.

Carroll, H., Beckstead, W., O'Connor, T., Ebbert, M., Clement, M., Snell, Q., and McClellan, D. (2007). DNA reference alignment benchmarks based on tertiary structure of encoded proteins. *Bioinformatics*, 23(19):2648–2649.

Churchill, G. A., von Haeseler, A., and Navidi, W. C. (1992). Sample size for phylogenetic inferencee. *Mol. Biol. Evol.*, 9:753–769.

Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25:1422–1423.

Dang, C. C., Le, Q. S., Gascuel, O., and Le, V. S. (2010). FLU, an amino acid substitution model for influenza proteins. *BMC Evol. Biol.*, 10:99.

Dimmic, M. W., Rest, J. S., Mindell, D. P., and Goldstein, R. A. (2002). rtREV: An amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny. *J. Mol. Evol.*, 55:65–73.

Douzery, E. J. P., Scornavacca, C., Romiguier, J., Belkhir, K., Galtier, N., Delsuc, F., and Ranwez, V. (2014). OrthoMaM v8: A database of orthologous exons and coding sequences for comparative genomics in mammals. *Molecular Biology and Evolution*, 31(7):1923–1928.

Dziak, J. J., Coffman, D. L., Lanza, S. T., Li, R., and Jermiin, L. S. (2020). Sensitivity and specificity of information criteria. *Brief. Bioinform.*, 21:553–565.

Felsenstein, J. (1978). The number of evolutionary trees. *Syst. Zool.*, 27:27–33.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.*, 17:368–376.

Felsenstein, J. (2004). *Inferring Phylogenies*. Sinauer Associates, Sunderland, Massachusetts.

Garland, T., Dickerman, A. W., Janis, C. M., and Jones, J. A. (1993). Phylogenetic analysis of covariance by computer simulation. *Systematic Biology*, 42:265–292.

Gascuel, O., editor (2005). *Mathematics of Evolution and Phylogeny*. Oxford University Press, Oxford, UK.

Gaut, B. S. and Lewis, P. O. (1995). Success of maximum likelihood phylogeny inference in the four-taxon case. *Molecular Biology and Evolution*, 12(1):152–162.

Graur, D. and Li, W.-H. (2000). *Fundamentals of Molecular Evolution*. Sinauer Associates, Sunderland, Massachusetts, 2 edition.

Gu, X., Fu, Y.-X., and Li, W.-H. (1995). Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. *Mol. Biol. Evol.*, 12:546–557.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conference*.

Hannaford, N. E., Heaps, S. E., Nye, T. M. W., Williams, T. A., and Embley, T. M. (2020). Incorporating compositional heterogeneity into Lie Markov models for phylogenetic inference. *The Annals of Applied Statistics*, 14(4):1964–1983, 20.

Hasegawa, M., Kishino, H., and Yano, T.-A. (1985). Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.*, 22:160–174.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919.

Hillis, D. M. (1995). Approaches for Assessing Phylogenetic Accuracy. *Systematic Biology*, 44(1):3–16.

Höhler, D., Pfeiffer, W., Ioannidis, V., Stockinger, H., and Stamatakis, A. (2022). RAxML grove: an empirical phylogenetic tree database. *Bioinformatics*, 38(6):1741–1742.

Huelsenbeck, J. P. and Hillis, D. (1993). Success of phylogenetic methods in the four-taxon case. *Syst. Zool.*, 42:247–264.

Johnson, J. B. and Omland, K. S. (2004). Model selection in ecology and evolution. *Trends Ecol. Evol.*, 19:101–108.

Jones, D. T., Taylor, W. R., and Thornton, J. M. (1992). The rapid generation of mutation data matrices from protein sequences. *Comput. Appl. Biosci.*, 8:275–282.

Jukes, T. H. and Cantor, C. R. (1969). Evolution of protein molecules. In Munro, H. N., editor, *Mammalian Protein Metabolism*, volume 3, pages 21–132. Academic Press, New York.

Kalyaanamoorthy, S., Minh, B. Q., Wong, T. K. F., von Haeseler, A., and Jermiin, L. S. (2017). Modelfinder: fast model selection for accurate phylogenetic estimates. *Nat. Methods*, 14:587–589.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795. doi: 10.1080/01621459.1995.10476572.

Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16:111–120.

Kimura, M. (1981). Estimation of evolutionary distances between homologous nucleotide sequences. *Proceedings of the National Academy of Sciences*, 78(1):454–458.

Kosiol, C. and Goldman, N. (2005). Different versions of the Dayhoff rate matrix. *Mol. Biol. Evol.*, 22:193–199.

Köster, J. and Sven, R. (2012). Snakemake – a scalable bioinformatics workflow engine. *Bioinformatics*, 28:2520–2522.

Kozlov, A. M., Darriba, D., Flouri, T., Morel, B., and Stamatakis, A. (2019). RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics*, 35(21):4453–4455.

Kupczok, A., Schmidt, H. A., and von Haeseler, A. (2010). Accuracy of phylogeny reconstruction methods combining overlapping gene data sets. *Algorithms Mol. Biol.*, 5:37.

Lanfear, R. (2019). Benchmarkalignments. https://github.com/roblanf/BenchmarkAlignments/. Accessed: 2023-04-26.

Le, S. Q. and Gascuel, O. (2008). An improved general amino acid replacement matrix. *Mol. Biol. Evol.*, 25:1307–1320.

Le, T. K. and Vinh, L. S. (2020). FLAVI: An amino acid substitution model for flaviviruses. *Journal of Molecular Evolution*, 88(5):445–452.

Le, V. S., Dang, C. C., and Le, Q. S. (2017). Improved mitochondrial amino acid substitution models for metazoan evolutionary studies. *BMC Evolutionary Biology*, 17(1):136.

Li, W.-H. (1997). *Molecular Evolution*. Sinauer Associates, Sunderland, Massachusetts.

McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*.

Minh, B. Q., Dang, C. C., Vinh, L. S., and Lanfear, R. (2021). QMaker: Fast and accurate method to estimate empirical models of protein evolution. *Syst. Biol.*, 70:1046–1060.

Minh, B. Q., Lanfear, R., Ly-Trong, N., Trifinopoulos, J., Schrempf, D., and Schmidt, H. A. (2022). IQ-TREE version 2.2.0: Tutorials and manual phylogenomic software by maximum likelihood. http://www.iqtree.org/doc/iqtree-doc.pdf. Accessed: 2023-04-26.

Minh, B. Q., Schmidt, H. A., Chernomor, O., Schrempf, D., Woodhams, M. D., von Haeseler, A., and Lanfear, R. (2020). Iq-tree 2: New models and efficient methods for phylogenetic inference in the genomic era. *Mol. Biol. Evol.*, 37:1530–1534.

Mölder, F., Jablonski, K. P., Letcher, B., Hall, M. B., Tomkins-Tinch, C. H., Sochat, V., Forster, J., Lee, S., Twardziok, S. O., Kanitz, A., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., and Köster, J. (2021). Sustainable data analysis with Snakemake. *F1000Res.*, 10:33.

Moretti, S., Laurenczy, B., Gharib, W. H., Castella, B., Kuzniar, A., Schabauer, H., Studer, R. A., Valle, M., Salamin, N., Stockinger, H., and Robinson-Rechavi, M. (2014). Selectome update: quality control and computational improvements to a database of positive selection. *Nucleic Acids Res*, 42(Database issue):D917–21.

Mount, D. W. (2004). *Bioinformatics: Sequence and Genome analysis.* Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 2. ed. edition.

Müller, T. and Vingron, M. (2000). Modeling amino acid replacement. *J. Comput. Biol.*, 7:761–776.

Nickle, D. C., Heath, L., Jensen, M. A., Gilbert, P. B., Mullins, J. I., and Kosakovsky Pond, S. L. (2007). HIV-specific probabilistic models of protein evolution. *PLoS One*, 2:e503.

Olsen, G. (1990). Interpretation of ”newick's 8:45” tree format. https://evolution.genetics.washington.edu/phylip/newick_doc.html. Accessed: 2023-04-24.

Posada, D. (2008). jModelTest: Phylogenetic model averaging. *Mol. Biol. Evol.*, 25:1253–1256.

Posada, D. and Buckley, T. R. (2004). Model selection and model averaging in phylogenetics: Advantages of akaike information criterion and bayesian approaches over likelihood ratio tests. *Systematic Biology*, 53(5):793–808.

Rota-Stabelli, O., Yang, Z., and Telford, M. J. (2009). MtZoa: A general mitochondrial amino acid substitutions model for animal evolutionary studies. *Mol. Phylogenet. Evol.*, 52:268–272.

Schmidt, H. A. (2003). *Phylogenetic Trees from Large Datasets.* PhD thesis, Universität Düsseldorf, Düsseldorf, Germany.

Schoch, C. L., Ciufo, S., Domrachev, M., Hotton, C. L., Kannan, S., Khovanskaya, R., Leipe, D., Mcveigh, R., O'Neill, K., Robbertse, B., Sharma, S., Soussov, V., Sullivan, J. P., Sun, L., Turner, S., and Karsch-Mizrachi, I. (2020). NCBI taxonomy: a comprehensive update on curation, resources and tools. *Database (Oxford)*, 2020.

Schwarz, G. (1978). Estimating the dimension of a model. *Ann. Stat.*, 6:461–464.

Sclove, S. L. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika*, 52(3):333–343.

Solomon, D. L. (1975). A note on the non-equivalence of the Neyman-Pearson and generalized likelihood ratio tests for testing a simple Null versus a simple alternative hypothesis. *The American Statistician*, 29(2):101–102. doi: 10.1080/00031305.1975.10477383.

Soubrier, J., Steel, M., S. Y. Lee, M., Der Sarkissian, C., Guindon, S., Ho, S. Y. W., and Cooper, A. (2012). The influence of rate heterogeneity among sites on the time dependence of molecular rates. *Mol. Biol. Evol.*, 29:3345–3358.

Stamatakis, A. (2014). RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313.

Tamura, K. and Nei, M. (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol. Biol. Evol.*, 10:512–526.

Tateno, Y., Takezaki, N., and Nei, M. (1994). Relative efficiencies of the maximum-likelihood, neighbor-joining, and maximum-parsimony methods when substitution rate varies with site. *Molecular Biology and Evolution*, 11(2):261–277.

Tavaré, S. (1986). Some probabilistic and statistical problems on the analysis of DNA sequences. In Miura, R. M., editor, *DNA Sequence Analysis (Proceedings of the 1984 Symposium Some Mathematical Questions in Biology)*, volume 17 of *Lectures on Mathematics in the Life Sciences*, pages 57–86. American Mathematical Society, Providence, Rhode Island, USA.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors (2007). *Workflows for e-Science*. Springer London.

Veerassamy, S., Smith, A., and Tillier, E. R. M. (2003). A transition probability model for amino acid substitutions from blocks. *Journal of Computational Biology*, 10(6):997–1010. doi: 10.1089/106652703322756195.

Wagenmakers, E.-J. and Farrell, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin & Review*, 11(1):192–196.

Whelan, S., de Bakker, P. I. W., Quevillon, E., Rodriguez, N., and Goldman, N. (2006). PANDIT: an evolution-centric database of protein and associated nucleotide domains with inferred trees. *Nucl. Acids Res.*, 34:D327–D331.

Woodhams, M. D., Fernández-Sánchez, J., and Sumner, J. G. (2015). A new hierarchy of phylogenetic models consistent with heterogeneous substitution rates. *Syst Biol*, 64(4):638–50.

Yang, Z. (1994). Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximative methods. *J. Mol. Evol.*, 39:306–314.

Yang, Z., Nielsen, R., and Hasegawa, M. (1998). Models of amino acid substitution and application to mitochondrial protein evolution. *Mol. Biol. Evol.*, 15:1600–1611.

Zharkikh, A. (1994). Estimation of evolutionary distances between nucleotide sequences. *J. Mol. Evol.*, 39:315–329.