Statistical Sequence Alignment of Protein Coding Regions

by

Juan José García Mesa

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2023 by the
Graduate Supervisory Committee:

Reed A. Cartwright, Chair
Jesse Taylor
Theodore Pavlic
Banu Ozkan

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

Sequence alignment is an essential method in bioinformatics and the basis of many analyses, including phylogenetic inference, ancestral sequence reconstruction, and gene annotation. Sequence artifacts and errors made in alignment reconstruction can impact downstream analyses, leading to erroneous conclusions in comparative and functional genomic studies. While such errors are eventually fixed in the reference genomes of model organisms, many genomes used by researchers contain these artifacts, often forcing researchers to discard large amounts of data to prevent artifacts from impacting results.

I developed COATi, a statistical, codon-aware pairwise aligner designed to align protein-coding sequences in the presence of artifacts commonly introduced by sequencing or annotation errors, such as early stop codons and abiological frameshifts. Unlike common sequence aligners, which rely on amino acid translations, only model insertion and deletions between codons, or lack a statistical model, COATi combines a codon substitution model specifically designed for protein-coding regions, a complex insertion-deletion model, and a sequencing base calling error step. The alignment algorithm is based on finite state transducers (FSTs), computational machines well-suited for modeling sequence evolution. I show that COATi outperforms available methods using a simulated empirical pairwise alignment dataset as a benchmark.

The FST-based model and alignment algorithm in COATi is resource-intense for sequences longer than a few kilobases. To address this constraint, I developed an approximate model compatible with traditional dynamic programming alignment algorithms. I describe how the original codon substitution model is transformed to build an approximate model and how the alignment algorithm is implemented by modifying the popular Gotoh algorithm. I simulated a benchmark of alignments and measured how well the marginal models approximate the original method.

Finally, I present a novel tool for analyzing sequence alignments. Available metrics can measure the similarity between two alignments or the column uncertainty within an alignment but cannot produce a site-specific comparison of two or more alignments. AlnDotPlot is an R software package inspired by traditional dot plots that can provide valuable insights when comparing pairwise alignments. I describe AlnDotPlot and showcase its utility in displaying a single alignment, comparing different pairwise alignments, and summarizing alignment space.

DEDICATION

Para Marise, José, Sebastián, y Samantha por su amor incondicional, por enseñarme a no rendirme nunca, y por siempre creer en mí.

Porque en esta vida todo pasa por algo.

ACKNOWLEDGMENTS

I want to thank the following people who supported me throughout my Ph.D. program. First, I would like to thank my advisor, Dr. Reed Cartwright, for his continuous mentorship, support, and guidance. Without his patience, this work would have been impossible.

I want to thank my committee members for improving my research and expanding my breadth of knowledge with their insights.

I want to thank the members of my lab, past and present, who provided me with feedback, inspired me with their enthusiasm, and created a cordial, pleasant, and intellectually stimulating environment to work in.

I want to thank the Biological Design Advising Staff for their help and kindness, the School of Life Sciences for the teaching assistant positions over the years, and the National Science Foundation for funding parts of this work.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Deoxyribonucleic acid (DNA) is considered the fundamental blueprint of life, with the primary purpose of storing and replicating the vital information required for the structure, function, and regulation within living organisms. In essence, DNA functions as the hard drive of nature, preserving the intricate details of the processes of life against data corruption events known as mutations. While cellular mechanisms exist to prevent and correct such errors, a fraction inevitably persists and eventually fixates in the population, contributing to the diversity of life through evolution.

Biological sequences are a collection of DNA molecules that carry genetic instructions across generations. DNA replicates itself and, through RNA, codes for proteins, which are responsible for virtually all chemical processes in organisms. The conservation of patterns in the three levels of biological information is paramount to enforce correct function preservation. Sequence analysis is the area of research within biology that studies the changes in DNA, RNA, and proteins.

Sequence alignment establishes a correspondence between the elements in a set of sequences that share a common ancestor and is the standard method to compare biological sequences. The alignment of DNA regions that code for proteins is typically performed in amino-acid space, which discards information, underperforms compared to alignment at the codon level, and can fail in the presence of artifacts. While some aligners incorporate codon substitution models, they do not support frameshifts or lack a statistical model. In addition, existing aligners typically force gaps to occur between codons, whereas in natural sequences, only about 42% of indels occur between codons (Taylor *et al.*, 2004; Zhu, 2022).

Throughout the following chapters, I describe the various intricacies and characteristics of models of molecular evolution, insertion and deletion (indel) models, and pairwise alignment algorithms. To do so, I must first set some definitions.

1.1 Notation

An **alphabet**, denoted by Σ, is a finite and unordered set of symbols. Let the DNA alphabet be defined

$$\Sigma_{DNA} = \{A, C, G, T\}$$

and the codon alphabet be all possible three-mers over the Σ_{DNA} alphabet, defined

$$\begin{aligned}\Sigma_{codon} = \{&AAA, AAC, AAG, AAT, ACA, ACC, ACG, ACT,\\ &AGA, AGC, AGG, AGT, ATA, ATC, ATG, ATT,\\ &CAA, CAC, CAG, CAT, CCA, CCC, CCG, CCT,\\ &CGA, CGC, CGG, CGT, CTA, CTC, CTG, CTT,\\ &GAA, GAC, GAG, GAT, GCA, GCC, GCG, GCT,\\ &GGA, GGC, GGG, GGT, GTA, GTC, GTG, GTT,\\ &TAA, TAC, TAG, TAT, TCA, TCC, TCG, TCT,\\ &TGA, TGC, TGG, TGT, TTA, TTC, TTG, TTT\}\end{aligned}$$

where, given a codon $X = \{X_1X_2X_3\} \in \Sigma_{codon}$, X_p denotes the nucleotide in position p of codon X. I often refer to symbols in Σ_{DNA} as nucleotides or residues and symbols in Σ_{codon} as codons for convenience. In addition, an alphabet can be extended to

include gaps $\Sigma \cup \{-\}$ or have symbols removed $\Sigma_{codon} - \{\text{TAA, TAG, TGA}\}$, in this case to indicate non-stop codons.

A **sequence** is a finite succession of the symbols in Σ. Given a sequence s of length m, I use $s_1 s_2 \cdots s_m$ to denote the symbols in s. The special case when a sequence contains no symbols is denoted $\varnothing$. In addition, let $|s|$ denote the length of sequence s. Furthermore, a **subsequence** of s is obtained by extracting zero or more symbols from s.

An **alignment** of sequences s and v is a two-row matrix A with entries in $\Sigma \cup \{-\}$ that meets three requirements: (i) the first and second row contain the symbols in s and v in order, respectively; (ii) one or more gaps $\{-\}$ are allowed between symbols in s and v; and (iii) every column contains at least one symbol in s or v, therefore a column comprised entirely of gaps is not allowed (Orlova, 2010). The alignment of two sequences is referred to as a pairwise alignment, whereas a multiple alignment contains three or more sequences.

Biologically, entries in an alignment can be seen as four events:

- No mutation: a residue remains unchanged.
- Substitutions: a residue is replaced by another.
- Insertions: a residue is added to a sequence in a specific position.
- Deletions: a residue is removed from a sequence.

1.2 Pairwise Sequence Alignment Algorithms

Needleman-Wunsch

Global pairwise alignment aims to find the optimal alignment of two sequences, providing a comprehensive view of their similarity. A classic global alignment algorithm is

the Needleman-Wunsch (NW) algorithm, which calculates the optimal alignment that maximizes the similarity of two sequences using dynamic programming (Needleman and Wunsch, 1970). Dynamic programming is used to build an optimal alignment using previous solutions for optimal alignments of smaller subsequences.

Algorithm 1 Needleman-Wunsch algorithm. Sequences s and v are aligned with gap penalty ϵ, and matching scoring function score().

```
1:  function NEEDLEMAN-WUNSCH(s, v, ϵ)
2:      n, m ← |s|, |v|
3:      M ← zero(n + 1, m + 1)        ▷ Initialize the matrix of size n + 1 by m + 1
4:      for i ← 1 to n do                                   ▷ Gap deletion penalties
5:          M[i, 0] ← i · ϵ
6:      end for
7:      for j ← 1 to m do                                   ▷ Gap insertion penalties
8:          M[0, j] ← j · ϵ
9:      end for
10:     for i ← 1 to n do                                   ▷ Fill in the matrices
11:         for j ← 1 to m do
12:             match ← M[i − 1, j − 1] + score(s_i, v_j)
13:             deletion ← M[i − 1, j] + ϵ
14:             insertion ← M[i, j − 1] + ϵ
15:             M[i, j] = max(match, deletion, insertion)        ▷ Save best score
16:         end for
17:     end for
18:     Traceback
19: end function
```

Given two sequences $s, v \in \Sigma$ with lengths m and n, respectively, the NW algorithm sets a matrix M with dimensions $m+1$ by $n+1$. The algorithm fills the matrix using a function `score()` that assigns values to matches and mismatches and gap cost (ϵ). Starting from the top-left to the bottom-right corner, row by row the score for each cell represents the optimal alignment score up to that position. The score for each cell is calculated by maximizing values from adjacent cells in three directions: diagonal for matches and mismatches, and up or down for indels. Algorithm 1 illus-

trates the NW algorithm. In addition, the algorithm performs a traceback operation to retrieve the optimal path and construct the aligned sequences.

Gotoh Algorithm

The Gotoh algorithm (Gotoh, 1982) is an extension of the NW algorithm that adds affine gap penalties. While the latter applies a constant penalty for each gap, the former applies an affine gap penalty model with different opening and extension gap penalties (i.e., a gap of length l, $score = open + extend \cdot (l-1)$). Affine gap penalties provide a more biological model where longer gaps are less penalized per column. In addition, the Gotoh algorithm can distinguish between insertions and deletions.

Algorithmically, this is translated into using three separate matrices to keep score of matches (M), deletions (D), and insertions (I). The matrix is filled similarly to the NW algorithm, row by row from the top-left to the bottom-right corner. Deletion and insertion openings originate from the match matrix, while gap extensions originate from their respective matrices (D and I). Algorithm 2 illustrates the Gotoh algorithm, with α and β as gap opening and extension costs, and the function `score()` that scores matches and mismatches. Note that the traceback algorithm is adapted to retrieve the optimal alignment through the tree matrices instead of one.

Algorithm 2 Gotoh pairwise alignment algorithm. Sequences s and v are aligned with an affine gap penalty with parameters α and β. Matches and mismatches are scored with the function score().

```
1: function ALIGNPAIR(s, v, α, β)
2:     n, m ← |s|, |v|
3:     M, D, I ← zero(n + 1, m + 1)  ▷ Initialize the matrices of size n + 1 by m + 1
4:     for i ← 1 to n do                                   ▷ Gap deletion penalties
5:         D[i, 0] ← α + (i − 1) · β
6:     end for
7:     for j ← 1 to m do                                   ▷ Gap insertion penalties
8:         I[0, j] ← α + (j − 1) · β
9:     end for
10:    for i ← 1 to n do                                   ▷ Fill in the matrices
11:        for j ← 1 to m do
12:            D[i, j] = max(M[i − 1, j] + α, D[i − 1, j] + β)
13:            I[i, j] = max(M[i, j − 1] + α, I[i, j − 1] + β)
14:            M[i, j] = max(M[i − 1, j − 1] + score(s_i, v_j), D[i, j], I[i, j])
15:        end for
16:    end for
17:    Add end weights
18:    Traceback
19: end function
```

Chapter 2

COATI: STATISTICAL PAIRWISE ALIGNMENT OF PROTEIN CODING SEQUENCES

2.1 Introduction

Sequence alignment is a fundamental task in bioinformatics and a cornerstone step in comparative and functional genomic analysis (Rosenberg, 2009). While sophisticated advancements have been made, the challenge of alignment inference has not been fully solved (Morrison, 2015). The alignment of protein-coding DNA sequences is one such challenge, and a common approach to this problem is to perform alignment inference in amino-acid space (e.g. Bininda-Emonds, Olaf 2005; Abascal *et al.* 2010). While this approach is an improvement over DNA models, it discards information, underperforms compared to alignment at the codon level, and fails in the presence of artifacts such as frameshifts and early stop codons. Although some aligners incorporate codon substitution models, they do not support frameshifts or lack a statistical model. In addition, existing aligners typically force gaps to occur between codons, whereas in natural sequences, only about 42% of indels occur between codons (Taylor *et al.*, 2004; Zhu, 2022). This mismatch between aligner assumptions and biology can produce sub-optimal alignments and inflated estimates of sequence divergence (Fig. 2.1).

Genome quality impacts conclusions drawn from comparative genomic studies, and uncorrected errors in the alignment stage can lead to erroneous results in comparative and functional genomic studies (Schneider *et al.*, 2009; Fletcher and Yang, 2010; Hubisz *et al.*, 2011). Genomes for model organisms often get refined over

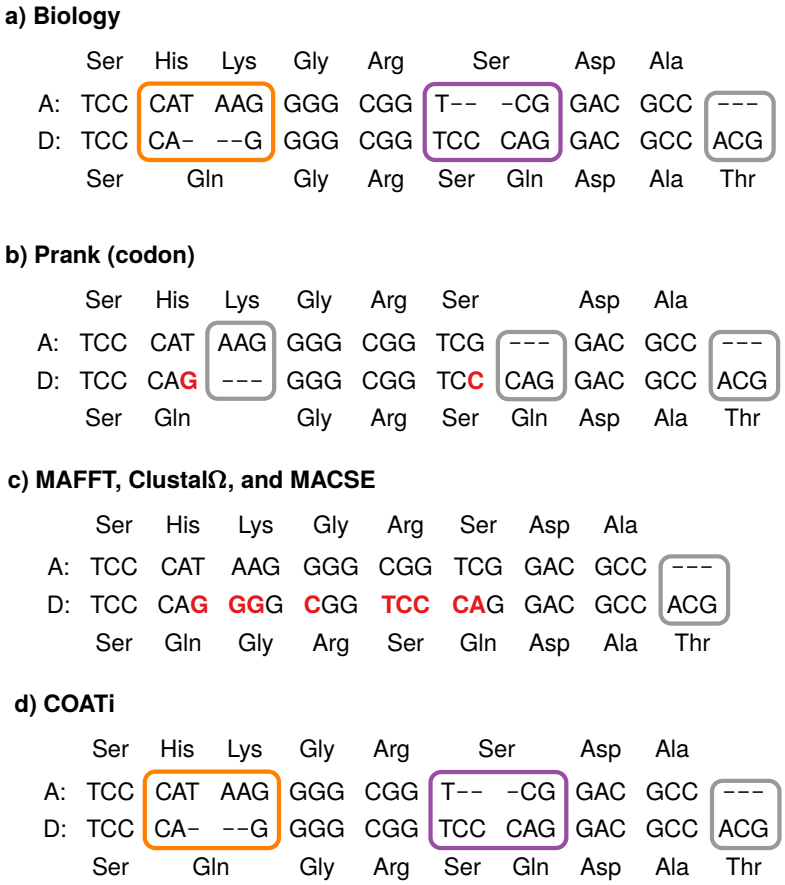


Figure 2.1: Standard algorithms produce suboptimal alignments. (a) shows the true alignment of an ancestor sequence (A) and a descendant sequence (D). (b), (c), and (d) are the results of different aligners. Nucleotide mismatches are highlighted in red. Phase 0, phase 1, and phase 2 indels are shown in gray, purple, and orange, respectively. Additionally, the orange indel is type II (an amino-acid indel plus an amino-acid change) while the purple indel is type I (an amino-acid indel only). COATi is the only aligner able to retrieve the biological alignment in this example.

many iterations and achieve high quality with meticulously curated protein coding sequences. In contrast, genomes for non-model organisms might only receive partial curation and typically have lower-quality sequences and annotations. These genomes often lack the amount of sequencing data needed to fix artifacts, including missing exons, erroneous mutations, and indels (Jackman *et al.*, 2018). When comparative and functional genomics studies include data from non-model organisms, care must

be taken to identify and manage such artifacts; however, current alignment methods are ill-equipped to handle common artifacts in genomic data, requiring costly curation practices that discard significant amounts of information. To address this problem, I developed COATi, short for COdon-aware Alignment Transducer, a pairwise statistical aligner that incorporates codon substitution models and is robust to artifacts present in modern genomic data.

2.2 Materials

2.2.1 Finite State Transducers

Statistical alignment is typically performed using pairwise hidden Markov models (pair-HMMs), which have the ability to rigorously model molecular sequence evolution (Bradley and Holmes, 2007). Pair-HMMs are computational machines with probabilities for emitting symbols from the states to two output tapes and probabilities for the transitions between the states (arcs). Each tape represents a sequence and a path through these computational machines is a possible pairwise alignment. Pair-HMMs contain a finite number of states, typically labeled match (M), insert (I), and delete (D). The emission probability distribution of M usually follows a substitution model for emitting an aligned pair or symbols (s_i, v_j) from sequences s and v. States I and D have distributions for emitting symbols against a gap (-). In addition, arcs in pair-HMMs have transition probabilities. Figure 2.2 illustrates a common pair-HMM architecture, with affine gap scoring. Conceptually, these machines generate two sequences, s and v, from an unknown ancestor and can calculate the probability that two sequences are related, represented by $P(s, v)$ (Yoon, 2009).

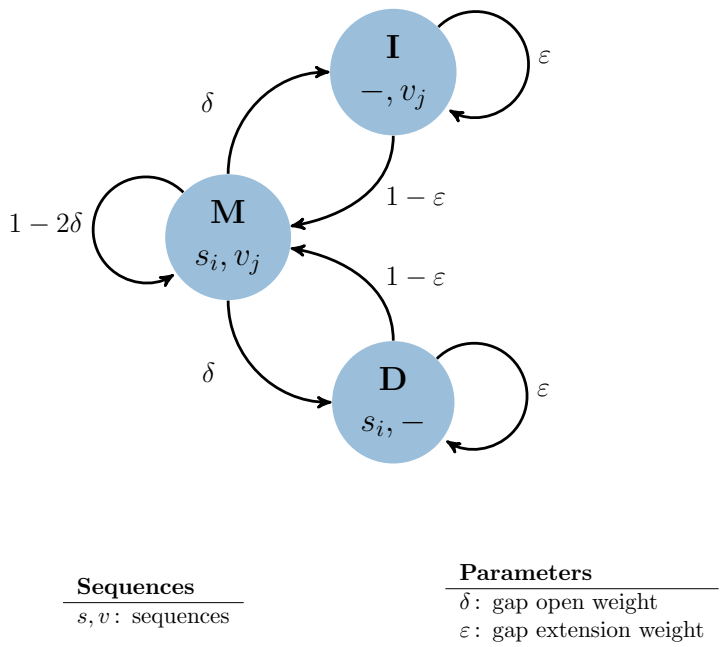


Figure 2.2: Example of a typical pair hidden Markov model (pair-HMM) used in statistical pairwise sequence alignment. Sequences s and v are aligned using an affine gap scoring model and a substitution model (unspecified). The arcs are weighted with gap opening parameter δ and gap extension ε, therefore defining no gap opening weight as $1 - \delta$ and gap closing as $1 - \varepsilon$.

A limitation of pair-HMMs is that they only model the evolution of two related sequences from an unknown ancestor. Finite-state transducers (FSTs) have similar benefits to pair-HMMs with the additional feature that they can model the generation of a descendant sequence given an ancestral one. FSTs, similar to pair-HMMS, are computational machines with a symbol alphabet, a set of states, and weighted arcs defining the transition probabilities between states. However, FSTs consume symbols from an input and emit symbols to an output tape ($a : b$), as opposed to having two output tapes (a, b). Properly weighted, an FST can calculate the probability that a descendant sequence, v, evolved from an ancestral sequence, s, represented by $P(v|s)$.

Furthermore, well-established algorithms for combining FSTs in different ways, including concatenation, composition, intersection, union, or reversal, allow the design

of complex models by combining simpler FSTs (Bradley and Holmes, 2007; Silvestre-Ryan *et al.*, 2021). Specifically, composition is a powerful and versatile algorithm for comparative sequence analysis, consisting of sending the output of one FST into the input of a second FST. Composition allows combining two or more FSTs to create a new, more complex transducer. Figure 2.3 illustrates how DNA transcription for a codon can be achieved by composing a complimenting FST with a transducer that replaces thymines with uracil, where the three nucleotides are read and complimented in 2.3-a, which are then used as the input of 2.3-b, resulting in the transcription of the codon (Fig. 2.3-c). COATi uses composition to derive a statistical alignment model from the combination of smaller FSTs, each representing a specific process.

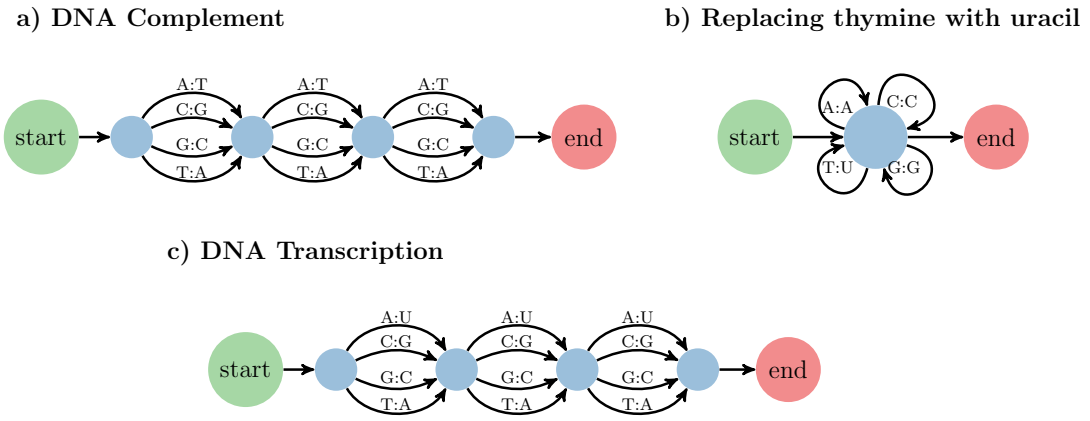


Figure 2.3: DNA transcription via FST composition. The composition of (a) a codon complimenting FST and (b) an FST that replaces T with U generates (c) an FST that implements codon transcription ($a \circ b = c$).

2.2.2 Evolution FST

COATi implements the pairwise alignment of a potentially lower-quality sequence against a high-quality sequence as a path through the Evolution FST (Fig. 2.4) (c.f. Holmes and Bruno, 2001). Here, COATi treats the high-quality (reference) sequence as the "ancestor" and the potentially lower-quality sequence as the "descendant". The

assumption is that the reference sequence is in-phase, which is used to help preserve the reading frame and safeguard against possible frameshifts in the "descendant". Therefore, the high-quality sequence must not contain incomplete codons (the number of nucleotides is multiple of three) and be free of any ambiguous nucleotides or early stop codons. In contrast, the potentially lower-quality sequence has no such restrictions and can be of any length, contain early stop codons—treated as artifacts—, and include ambiguous codons.

The Evolution FST is the result of composing a substitution FST that encodes a codon model (Fig. 2.4-a), an indel FST that models insertions and deletions, including frameshifts (Fig. 2.4-b), and a base calling error FST that models incorrectly sequenced bases (Fig 2.4-b). A key innovation of this FST, with respect to others, is the combination of a codon substitution model with a nucleotide-based geometric indel model that allows gaps to occur at any position.

Composing both sequences with the Evolution FST results in the transducer of all possible alignments. Any path through this FST represents a pairwise alignment, while the shortest path (by weight) corresponds to the best alignment. All FST operations in COATi, including model development, composition, search for the shortest path, and other optimization algorithms, are performed using the C++ openFST library (Allauzen *et al.*, 2007). However, the Evolution FST has a large state space to keep track of codon substitution rates when codons might be interspersed with indel events. This additional state space increases the computational complexity of the alignment algorithm.

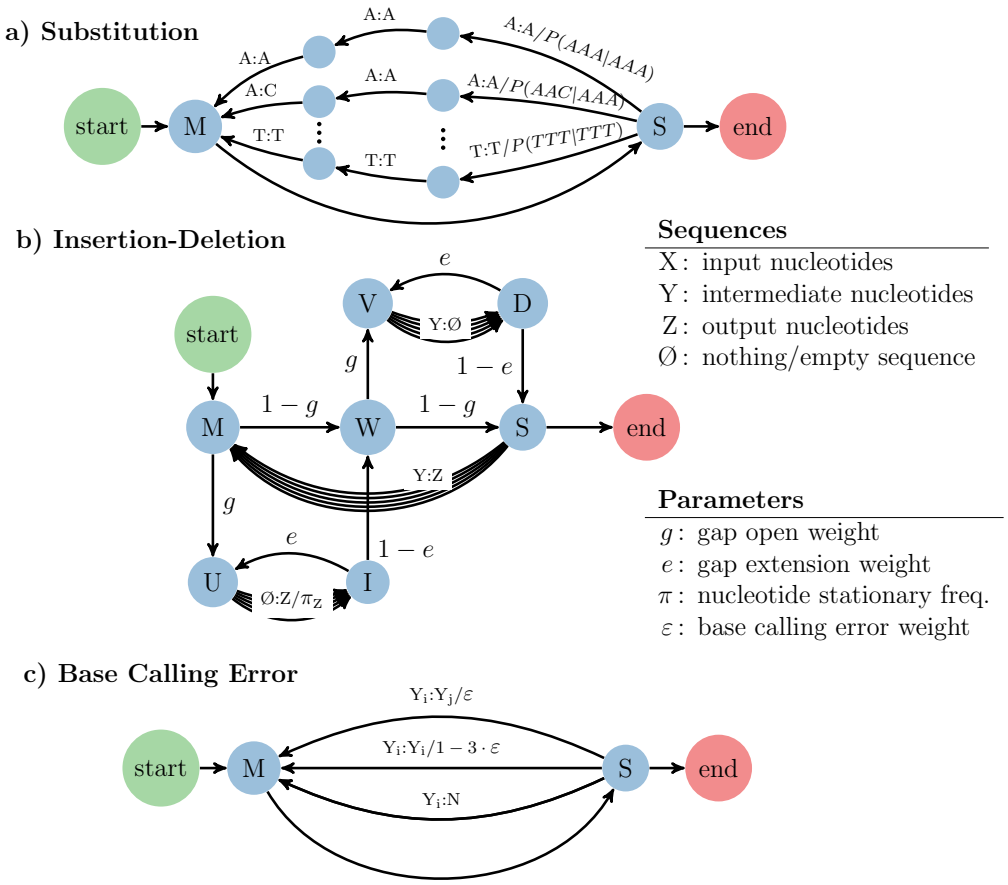


Figure 2.4: The Evolution FST is assembled by composing a substitution FST and an indel FST. Each node represents a state in an FST while arcs display possible transitions between states (and their weights). The arc label format is input symbol : output symbol / weight. Unlabeled arcs have weights of 1, and partially labeled arcs do not consume/emit symbols or have a weight of 1. (a) The substitution FST encodes a 61×61 codon substitution model with 3721 arcs from S to M. These arcs consume three nucleotides from the input tape and emit three nucleotides to the output tape. The weight of each arc is a conditional probability derived from a codon substitution model. (b) The indel FST allows for insertions (U to I) and deletions (V to D). Insertion arcs are weighted according to the codon model's stationary distribution of nucleotides, and deletion arcs have a weight of 1. Contiguous insertions and deletions are always arranged for insertions to precede deletions to limit equivalent alignments. The base calling error FST (c) is added on top of the indel FST to model sequencing errors. Arcs from S to M generate matches; however, with (c) they can introduce single-nucleotide errors, which allows modeling stop codon artifacts.

2.2.3 Codon Substitution Models

Muse and Gaut

Codon substitution is particularly suitable for modeling protein-coding genes since it accounts for both the likelihood of mutations occurring at the nucleotide level and selective pressure on amino acid substitutions (Sullivan and Joyce, 2005). While codon models are extensively applied to reconstruct phylogenies and study natural selection (Delport *et al.*, 2009), their use in alignment reconstruction is still scarce. COATi is the only aligner that implements the mechanistic Muse and Gaut model (MG94), a codon model designed for coding regions (Muse and Gaut, 1994). Modeled as a continuous-time Markov process, the instantaneous substitution rate matrix Q defines the rate that codon X changes to codon Y as

$$Q_{XY} = \begin{cases} 0 & \text{if } X \text{ and } Y \text{ differ by more than one nucleotide change} \\ \mu_{X_pY_p} & \text{if } X \text{ and } Y \text{ are synonymous} \\ \omega \cdot \mu_{X_pY_p} & \text{if } X \text{ and } Y \text{ are nonsynonymous} \end{cases} \tag{2.1}$$

$$Q_{XX} = -\sum_{Y:Y \neq X} Q_{XY} \tag{2.2}$$

where X and Y are non-stop codons defined $X, Y \in \Sigma_{codon} - \{\text{TAA, TAG, TGA}\}$, ω is the strength of selection affecting amino-acid changes, and X_p and Y_p refer to the nucleotides in position p in codons X and Y respectively, defined $X_p, Y_p \in \Sigma_{DNA}$. $\mu_{X_pY_p}$ represents the mutation rate that nucleotide X_p is replaced by Y_p defined $\mu_{X_pY_p} = \pi_{Y_p}\sigma_{X_pY_p}, \forall\ X_p \neq Y_p$, where π_{Y_p} is the equilibrium frequency of nucleotide Y_p, and $\sigma_{X_pY_p}$ corresponds to one of the instantaneous substitution parameters (Table 2.1) of the General Time Reversible nucleotide substitution model (Tavaré *et al.*, 1986)

(GTR). I use the nucleotide stationary frequencies and GTR parameters inferred from primate data by Yang 1994 to construct MG94 in COATi.

The GTR model is a well-known and widely-used substitution model that allows different instantaneous mutation rates between each of the six nucleotide pairs, with equal forward and reverse rates for any given pair. This property is represented $\pi_{Y_p}\mu_{X_pY_p} = \pi_{X_p}\mu_{Y_pX_p}$ and transfers to the MG94 model, making both models time-reversible.

	A	C	G	T
A	$*$	$\pi_C\sigma_{AC}$	$\pi_G\sigma_{AG}$	$\pi_T\sigma_{AT}$
C	$\pi_A\sigma_{CA}$	$*$	$\pi_G\sigma_{CG}$	$\pi_T\sigma_{CT}$
G	$\pi_A\sigma_{GA}$	$\pi_C\sigma_{GC}$	$*$	$\pi_T\sigma_{GT}$
T	$\pi_A\sigma_{TA}$	$\pi_C\sigma_{TC}$	$\pi_G\sigma_{TG}$	$*$

Table 2.1: Instantaneous nucleotide substitution rates for the General Time Reversible model (GTR). On each row, the parameters represent the probability of a given nucleotide being replaced by another. GTR is the most general and time-reversible nucleotide substitution model, with a different mutation rate parameter for each of the six possible nucleotide combinations. Note that sigma parameters for each of the six possible nucleotide pairs are identical (i.e., $\sigma_{AC} = \sigma_{CA}$, $\sigma_{AG} = \sigma_{GA}$, etc.).

Empirical Codon Model

In addition to the MG94 codon model, COATi incorporates an empirical codon model that can be used with the triplet and marginal alignment procedures. While mechanistic models explicitly account for molecular evolution features and use a defined set of parameters to specify them, empirical models, in contrast, attempt to summarize the substitution patterns inferred from extensive datasets. Although codon substitution models are rare in sequence aligners, the Empirical Codon Model (ECM) is the most common. This model is characterized by incorporating instantaneous doublet and triplet changes and encoding the physicochemical properties of amino acids (Kosiol *et al.*, 2007). The ECM model was estimated using 7,332 protein families from the

Pandit database, a collection of protein-coding multiple sequence alignments (Whelan *et al.*, 2006).

Although purely empirical substitution models can be applied as is, the empirical codon model offers a combined approach with mechanistic parameters. Similar to the MG94 model definition, the instantaneous substitution rate matrix Q for the ECM model defines the rate that codon X changes to codon Y as

$$Q_{XY} = \begin{cases} s^*_{XY} \cdot \pi_Y & \text{if } X \text{ and } Y \text{ are synonymous} \\ s^*_{XY} \cdot \pi_Y \cdot \omega & \text{if } X \text{ and } Y \text{ are nonsynonymous} \end{cases} \tag{2.3}$$

$$Q_{XX} = -\sum_{Y:Y \neq X} Q_{XY} \tag{2.4}$$

where X and Y are non-stop codons defined $X, Y \in \Sigma_{codon} - \{\text{TAA, TAG, TGA}\}$, s^*_{XY} are the ECM exchangeabilities estimated from the Pandit database, and π is the frequency of codon Y. Note that this model is also time-reversible.

For both MG94 and ECM, the probability that codon Y replaces codon X after time t is calculated via matrix exponentiation: $P(Y|X;\Theta) = (e^{Qt})_{XY}$, where Θ is the set of models parameters $\Theta_{MG} = \{t, \omega, \pi, \sigma\}$ for MG94 and $\Theta_{ECM} = \{t, \omega, \pi, s^*\}$ for ECM (Holmes and Rubin, 2002). Note that as these models are applied in the context of protein-coding sequences, stop codons are not considered, resulting in a 61x61 matrix. In addition, probabilities are log-transformed to prevent underflow.

Codon substitution models are uncommon in sequence aligners, despite their extensive use in phylogenetics. COATi implements the Muse and Gaut (1994) codon model (codon-triplet-mg) and the Empirical Codon Model (Kosiol *et al.*, 2007) (codon-triplet-ecm). It also lets the user provide a codon substitution matrix. The default FST model (codon-triplet-mg) does not allow early stop codons in the ancestor sequence; although, it does support mutations to (early) stop codons under the assump-

tion that these are artifacts common in low-quality data.

Approximate Substitution Model

To reduce the runtime complexity of COATi, I have also developed an approximation of the Evolution FST that can be implemented with standard dynamic programming techniques. This approximation uses a marginal substitution model where the output nucleotides are independent of one another and only depend on the input codon and position. This produces a $(61 \times 3) \times 4$ substitution model and eliminates the need to track dependencies between output nucleotides.

If we let $X = \{X_1X_2X_3\}$ and $Y = \{Y_1Y_2Y_3\}$ be codons from Σ_{codon}, composed by nucleotides $\{X_1, X_2, X_3\} \in \Sigma_{DNA}$ and $\{Y_1, Y_2, Y_3\} \in \Sigma_{DNA}$ respectively, the probability that the descendant codon Y substitutes the ancestral codon X after time $t \in \Theta$ by the triplet model is $P\left(Y_1Y_2Y_3|X_1X_2X_3;\Theta\right)$. Then, we define the marginal model substitution probability that the nucleotide X_p changes to nucleotide $y \in \Sigma_{DNA}$ as

$$P_{\text{mar}}\left(Y_p = y|X_1X_2X_3;\theta\right) = \sum_{Y_1Y_2Y_3} I(Y_p = y)P\left(Y_1Y_2Y_3|X_1X_2X_3;\Theta\right) \tag{2.5}$$

where θ contains the marginal model parameters and is defined $\theta = \Theta \cup p$, p represents the position of the descendant nucleotide relative to the ancestral reading frame, defined $p \in \{1, 2, 3\}$, and $y \in \Sigma_{DNA}$ is the descendant nucleotide. Additionally, I is an indicator function that returns one if the left-hand side and right-hand side nucleotides are equal $I(e) = \{1 \text{ if } e \text{ is true and } 0 \text{ otherwise}\}$.

COATi contains marginal models for both MG94 and ECM, resulting in the marginal models codon-marginal-mg and codon-marginal-ecm. These models emphasize the position in a codon where the substitution occurs, help restrict the effects of low-quality data in the descendant sequence, and allow more than one substitution

per codon. In combination with the indel model, alignment using the marginal model is implemented using dynamic programming. Here, I provide a brief introduction to the marginal model and later evaluate its accuracy alongside the triplet model. However, a detailed description of the design and implementation of the approximate model is provided in the following chapter 3.

2.3 Methods

2.3.1 Empirical Simulation Algorithm

Simulating sequence evolution plays an essential role in bioinformatics, as an indispensable tool in validating novel methods, evaluating the performance of phylogenetic methods, and testing hypotheses among others (Ly-Trong *et al.*, 2022). In sequence alignment, benchmark datasets are frequently used for testing alignment algorithms and estimating model parameters under different evolutionary conditions. Sequence simulation algorithms are typically used when knowing the true parameter values of the underlying is required. There exists a wide array of DNA sequence alignment simulators such as DAWG (Cartwright, 2005), INDELible (Fletcher and Yang, 2009), and AliSim (Ly-Trong *et al.*, 2022) that can mimic evolutionary phenomena using a variety of parameter-rich models. However, when it is not required to know the true parameters that govern the benchmark, empirical data can yield a more accurate assessment.

While datasets of curated amino acid multiple sequence alignments available for tool validation are limited (e.g., Thompson *et al.* (2005); Raghava *et al.* (2003)), they are non-existing for DNA sequences, especially for pairwise alignments. Therefore, I developed an empirical simulation algorithm to compare the performance of COATi against commonly used sequence aligners. I downloaded 16000 human genes and

their gorilla orthologs from the ENSEMBL database (Hubbard *et al.*, 2002). After downloading, I removed 2232 sequence-pairs longer than 6000 nucleotides and aligned the remaining pairs with all five methods. At least one aligner added gaps to 6048 sequence pairs, and no aligner added gaps to 7719 sequence pairs. Then, I randomly introduced gap patterns extracted from all five methods into the ungapped sequence pairs to generate the benchmark alignments.

The simulation algorithm can introduce a pairwise alignment pattern to any two nucleotide sequences of equal length. The alignment pattern is given as a CIGAR string (Compact Idiosyncratic Gapped Alignment Report), a format commonly used to summarize aligned reads to a reference genome. Assigning one of the sequences as the reference, to distinguish between insertions and deletions, CIGAR strings can also summarize pairwise alignments by grouping the number of contiguous matches or mismatches 'M', deletions 'D', and insertions 'I'. The resulting pattern combines these letters preceded by the number of characters for each section as they appear in the alignment. This pattern is introduced by replacing nucleotides with gaps as indicated by deletions on one sequence and randomly introducing residues where the CIGAR strings indicated insertions.

Several safety checks are in place to ensure the algorithm runs correctly and the result is accurate. The assertions are divided into checking lengths and maintaining the reading frame of each section. The simulation can fail if the length of the sequences is different or if, without counting insertions, the length of the pattern to be inserted is longer. In addition, maintaining the phases of each section in the CIGAR string is important to avoid introducing errors such as frameshifts.

I created the benchmark of alignments by using an equal number of randomly sampled gap patterns from each aligner. I used the dataset to evaluate the accuracy of COATi and a suite of popular aligners spanning various alignment methods: ClustalΩ

v1.2.4 (Sievers *et al.*, 2011), MACSE v2.06 (Ranwez *et al.*, 2011), MAFFT v7.505 (Katoh and Standley, 2013), and PRANK v.150803 (Löytynoja, 2014).

2.3.2 Metrics

To quantify the similarity between each alignment in the benchmark and the corresponding output obtained by the different tools, I used the alignment error metric d_{seq} (Blackburne and Whelan, 2011). This metric accounts for indels and is more informative than conventional distance scores like sum-of-pairs or total columns. Intuitively, d_{seq} ranges between zero and one and can be interpreted as the probability that a randomly selected residue will be aligned to a different location against a sequence that does not contain such residue.

In addition, I compared the number of perfectly and imperfectly retrieved alignments for each aligner. Perfect alignments are defined as those with a distance of zero to the reference alignment ($d_{seq} = 0$), indicating 100% similarity. Notably, a set of sequences can have more than one optimal alignment under the same evolutionary model (same score), despite algorithms typically producing a single result. Consequently, to account for evolutionary equivalent alignments, I scored all alignments using the marginal model and considered perfect those with scores identical to the benchmark. Furthermore, I counted the number of alignments with the lowest distance d_{seq} to the true alignment, including ties, reported as best alignments. Moreover, I computed the count of imperfect alignments, where an alignment is considered imperfect when its distance to the reference alignment is greater than zero ($d_{seq} > 0$) and another method successfully produced an alignment with 100% similarity. This analysis exposes instances where all aligners fall short of achieving a perfect result in addition to a direct comparison.

To evaluate how well the aligners were able to identify positive and negative

selection, I estimated k_s and k_a statistics. k_s and k_a are, respectively, the number of substitutions per synonymous site (no changes at the amino acid level) and per non-synonymous site (introduces changes at the amino acid level) between two protein-coding genes. They are also denoted as d_s and d_n in the literature. I used the R package seqinr (Charif and Lobry, 2007) to estimate these metrics, which follows the popular method put forth by Li (Li, 1993). First, this method takes two aligned homologous protein-coding sequences and classifies the nucleotide sites in a sequence as nondegenerate, twofold degenerate, and fourfold degenerate. A site is nondegenerate if all possible changes at that site are nonsynonymous, twofold degenerate if one of the three possible changes is synonymous, and fourfold degenerate if all possible changes are synonymous. Second, the nucleotide changes between the two sequences are counted and divided as transitional (A↔G, C↔T) and transversional ({A, G}↔{C, T}). Third, the Kimura two-parameter distance is used to estimate the number of transitions and transversions per site type (nondegenerate, twofold degenerate, and fourfold degenerate), which is used as a correction factor for multiple hits. Finally, k_s is the estimate of the average transitional rate at twofold and fourfold degenerate sites, and k_a is the estimate of the average transversional rate at nondegenerate and twofold sites. In the results, these metrics are reported as the F_1 score, which is the harmonic mean of precision (true positives over total positives) and recall (true positives over true positives and false negatives). This score ranges between 0 and 1, with a score of 1 representing a perfect result.

2.4 Results

COATi, using the codon-triplet-mg model, obtained better results compared to a wide variety of alignment strategies. It was significantly more accurate (lower d_{seq}) at inferring the empirically simulated alignments compared to other methods; all p-

values were less than $2.1 \cdot 10^{-79}$ according to the one-tailed, paired Wilcoxon signed-rank tests (Supplementary Materials Figure 1). In addition, COATi produced more perfect alignments, less imperfect alignments, and more accurately inferred events of positive and negative selection (Table 2.2).

	COATi	**MAFFT**	**PRANK***	**MACSE**	**ClustalΩ**
Method	Trip-MG	DNA	Codon	DNA+AA	AA
Distance metric d_{seq}	0.00221	0.01471	0.01828	0.01399	0.02929
Best alignments	5139	4692	4774	3737	2615
Perfect alignments	5793	5292	4725	2861	2893
Imperfect alignments	1048	1549	2116	3980	3948
F1 positive selection	98.1%	84.3%	86.7%	79.5%	68.7%
F1 negative selection	99.8%	98.4%	98.7%	98.2%	96.9%

*PRANK produced 50 empty alignments, calculations are based on 7669 alignments.

Table 2.2: COATi generates better alignments than other alignment algorithms. Results of COATi, PRANK, MAFFT, ClustalΩ, and MACSE aligning 7719 empirically simulated sequence pairs. Best alignments have the lowest d_{seq} (including ties), perfect alignments have the same score as the true alignment, and imperfect alignments have a different score than the true alignment when at least one method found a perfect alignment.

ClustalΩ generated alignments via amino acid translations and obtained the highest average alignment error while having difficulties retrieving positive selection. MACSE used a DNA-AA hybrid model, allowing frameshifts, and obtained similar results to MAFFT using a DNA model. PRANK, using a codon model, had an average alignment error between MACSE/MAFFT and ClustalΩ but was unable to generate alignments for some sequence pairs.

In addition, I repeated the analysis with equal parameters to test the codon-triplet-ecm model (Tab. A.2) and the marginal model (Tab. A.3, A.4). In all cases, COATi outperformed all other methods in all metrics. Results are available in appendix A.

2.4.1 Ancestor and Descendant Sequences

To test how well COATi performs when the roles of reference and low-quality sequence are reverted, I aligned the 7761 simulated alignments using gorilla as the reference. Notably, COATi was only able to align 4003 sequence pairs due to the presence of early stop codons in the gorilla sequence on the remaning alignments. While the simulation algorithm prevents disrupting the reading frame and introducing frameshifts, it does not prevent early stop codons from being formed in the descendant sequence. Despite this limitation, I analyzed the 4003 alignments and compared the results with all methods, including COATi using the human sequence as the reference. The results (Tab. A.5) show a decrease, albeit small, in accuracy across all metrics when the low-quality sequence is used as the ancestor in comparison to the reverse. However, these results continue to be a significant improvement over other aligners.

2.5 Discussion

Despite human and gorilla sequences having a relatively short evolutionary distance, COATi showed a biologically significant improvement over other methods, with an average alignment error nine-fold smaller than the next best method. COATi is an FST-based application that can calculate the optimal alignment between a pair of sequences in the presence of artifacts using a statistical model. Using COATI will allow researchers to analyze more data with higher accuracy and facilitate the study of important biological processes that shape genomic data.

The models in COATi, as is inherent to all models in biology, aim to approximate the evolutionary processes that take place in nature and, therefore, have limitations. An assumption in the pairwise aligner is directionality in evolution. Specifically, one sequence is treated as the ancestor, while the other assumes the role of the

descendant. This assumption stems from the premise that the ancestor sequence is of higher quality, which the model leverages to preserve the reading frame and eliminate potential artifacts in the descendant sequence.

Although this characteristic of the model benefits the accuracy of the alignment, as it filters out errors in sequencing and annotation, it introduces a bias. For the case between human and gorilla, reversing the roles does not substantially impact the results. However, I propose two potential solutions to mitigate the ancestor-descendant bias. A straightforward approach that can be applied to large datasets, where the goal is to compute summary statistics, is to assign the ancestor role to either sequence. Alternatively, a more robust solution is to modify the alignment algorithm to conduct two Viterbi runs, using a different sequence as the ancestor each time and finding the path that maximizes both Viterbi tracebacks.

Future work also includes extending the indel FST to combine a 3-mer gap model with a frameshift parameter and weighing each indel phase differently to reflect known selection on indel phases (Zhu, 2022).

2.6 Availability

The source code for COATi, along with documentation, is freely available on GitHub: `https://github.com/CartwrightLab/coati` and is implemented in C++. Additional information, code, and workflows to replicate the analysis can be found on GitHub: `https://github.com/jgarciamesa/coati-testing`.

Chapter 3

A COMPARATIVE STUDY OF EVOLUTIONARY MODELS IN COATI

3.1 Introduction

Sequence alignment plays a pivotal role in most bioinformatic analyses by providing a fundamental framework for the analysis of evolution, the central driving force in biology (Aniba *et al.*, 2010). Computational biologists frequently address evolutionary questions such as phylogenetic inference, ancestral sequence reconstruction, identification of disease-associated mutations, and measurement of selection through the analyses of genomic data, which require the use of alignment inference (Rosenberg, 2009). Alignments of sequences are not observed directly and must be inferred from sequence data using algorithms. Specifically, a sequence alignment is a hypothesis of which characters in two or more sequences are related by common descent (Cartwright, 2006). Every stage within a genomics analysis pipeline is vulnerable to errors, including those that precede sequence alignments, such as DNA contamination, sequencing and assembly errors, or misannotations of genomes (Zhang *et al.*, 2021). Consequently, sequence aligners face the challenge of accounting for these artifacts to prevent uncorrected errors from producing misleading results in comparative and functional genomic studies (Schneider *et al.*, 2009; Fletcher and Yang, 2010; Hubisz *et al.*, 2011).

In the previous chapter, I presented COATi, a statistical pairwise aligner that understands and corrects common genomic artifacts. Unlike other statistical aligners, COATi is based on finite state transducers (FSTs) and harnesses the inherent properties of FSTs to model features of molecular evolution. COATi combines a reversible

codon substitution model, an insertion-deletion (indel) model that understands gap phases and handles frameshifts, and a step that models sequencing base calling errors that can introduce early stop codons. Results show that COATi is more accurate than current tools when aligning protein-coding sequences, especially in the presence of artifacts such as early stop codons or frameshifts.

The design and implementation of novel algorithms are instrumental in advancing scientific understanding and enabling groundbreaking discoveries. In the field of genomics, where the magnitude of data in modern analyses is staggering, accuracy and speed are crucial prerequisites for developing innovative methods. This necessity for efficiency is the motivation behind integrating an approximate evolutionary model into COATi. While the triplet model in COATi, with an FST-based alignment algorithm, outperforms other available tools in terms of accuracy, the computational complexity of its implementation can become a bottleneck when processing lengthy sequences. To address this challenge, I introduced a marginal approximation to the evolutionary model in the previous chapter. This approximation harnesses traditional dynamic programming techniques, enhancing the capabilities of COATi to meet the demands of modern genomics.

In the upcoming sections of this chapter, I examine the relationship between the approximate and triplet models. While results show both approaches perform significantly better than other aligners (as detailed in Appendix A), here I provide a comprehensive description of the approximate models and clarify how well they estimate the triplet model through a detailed analysis of relevant metrics.

3.2 Materials

3.2.1 Indel Model

The triplet model in COATi is created by composing three FSTs, each tailored to represent a specific process of molecular evolution (substitutions and indels) or designed for error correction (base calling error). Specifically, the indel FST models insertions and deletions and assumes that the length of each deleted or inserted region has a geometric distribution (Fig. 3.1). In addition, contiguous indels are exclusively modeled by insertions preceding deletions, with a direct path from the insertion state (I) to the deletion state (D) through the intermediate state (W), while the reverse path (D to I) must go through the match state (M). This design choice effectively limits the number of equivalent alignments and results in a more efficient alignment for the triplet model. Notably, the order of contiguous indels (i.e., insertion followed by deletion or deletion followed by insertion) does not affect the alignment score in this model.

Given the inherent characteristics of the triplet codon substitution model, the alignment can only be performed using FST-based algorithms, which can become costly for long pairs of sequences. To enhance COATi's data processing capacity and broaden its usability, I developed an approximate model that facilitates a faster alignment implementation. This is achieved by transforming the codon substitution probability matrix, thus allowing the reduction of the triplet model to a three-state FST (Fig. 3.2). This streamlined model preserves its original indel weights and is compatible with established pairwise dynamic programming methods.

As described in the previous chapter, the triplet evolutionary FST model calculates the probability of a descendant sequence given an ancestral one, with parameters branch length, coefficient of selective pressure, and residue stationary frequencies.

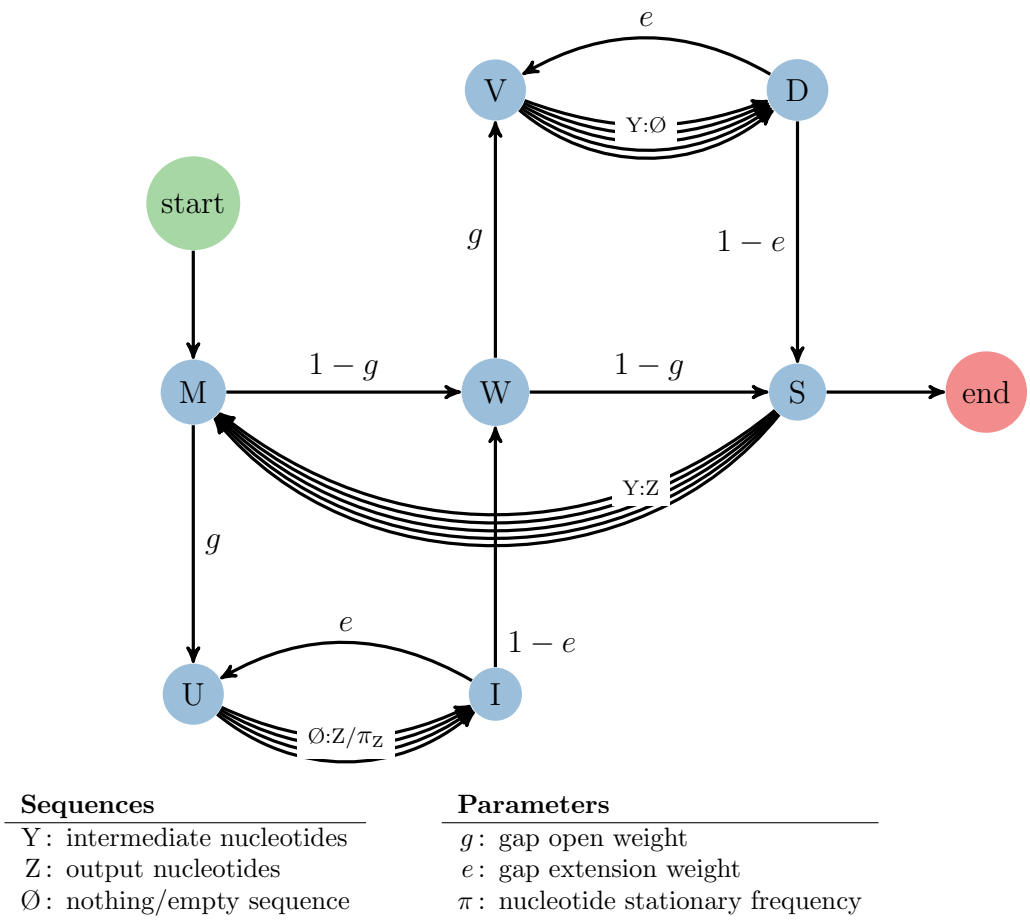


Figure 3.1: Indel FST for the triplet model. Allows insertions (U to I) and deletions (V to D). Insertion arcs are weighted by the corresponding nucleotide stationary frequency (π). The path and weight to the end state are designed so that the start and end gaps are symmetric. The arc weights are defined in linear space using gap opening (g) and gap extension (e) parameters. Arcs indicate an input symbol : output symbol / and weight. Missing symbols indicate no consumption or emission of symbols and missing weights indicate a weight of 1.

This probability is conditional on starting and ending the alignment with a pseudo-match to simplify the model and for start and end gaps to be symmetric, which results in the alignment weight being affected by a scaling factor. While this does not affect the ability of COATi to find the best alignment, this condition is inherited by the approximate model.

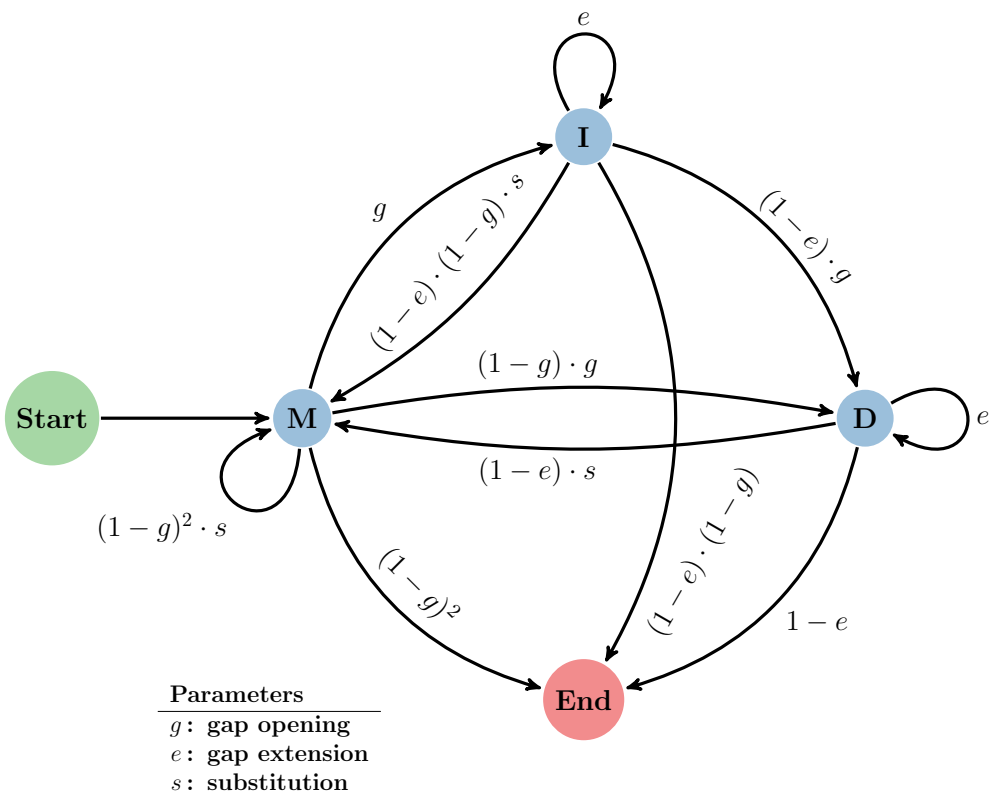


Figure 3.2: Approximate indel model, a three-state pairwise alignment architecture based on the Evolution FST described in COATi. Arcs to the match state (M) take and emit nucleotides based on a modified codon substitution model probability, whereas arcs to indel states introduce insertions (I) or deletions (D). The transition probabilities are defined in linear space and weighted using parameters gap opening (g), gap extension (e), and substitution (s).

3.2.2 Alignment and Semirings

Pairwise statistical alignment defines a probabilistic framework for finding the minimum cost or the most likely path through a state transition graph, where each state represents a position in two biological sequences, and each transition represents an evolutionary event (no substitution, substitution, insertion, or deletion). The optimal path corresponds to the most biologically meaningful alignment of the sequences under a specific model. Since the transition probabilities are generally less than one, their product can result in small numbers, especially when dealing with long DNA

sequences. Due to the limitations of standard floating-point arithmetic in modern computers, this can lead to numerical underflow. When this occurs, the likelihood can become virtually zero, thus leading to incorrect results. To address this issue, calculations in statistical alignment algorithms are typically performed in logarithmic space, thus preventing underflow (Durbin *et al.*, 1998).

Semiring	Set	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Linear	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	-∞	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	max	$+$	-∞	0

Table 3.1: Types of semirings implemented in COATi and their defined operations.

COATi includes two alignment procedures, one based on FST operations for the triplet model and one based on Gotoh's (1982) algorithm for the approximate model (Appendix B). Despite having different implementations, both approaches perform their computations in logarithmic space through the use of semirings, either as included in the openFST library (Allauzen *et al.*, 2007) for the former or self-implemented for the latter. Mathematical semirings are algebraic structures that define two binary operations, usually denoted by addition ($\oplus$) and multiplication ($\otimes$). Addition behaves like a commutative monoid with identity element $\overline{0}$ ($a \oplus b = b \oplus a$ and $a \oplus \overline{0} = a$), while multiplication is distributed over addition ($a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$) and has an identity element $\overline{1}$ ($a \otimes \overline{1} = a$). In particular, the tropical semiring (Table 3.1) is ideal for implementing the Viterbi algorithm with logarithmic transition scores, which finds the best path through a sequence of states by calculating the score of each possible transition for each state using the $\otimes$ operator ($+$) and selects the optimal one with $\oplus$ (max). Both alignment procedures are implemented using the tropical semiring.

3.2.3 Marginal Substitution Model

I developed the approximate model to speed up the alignment in COATi. The main restriction of the triplet model that prevents its implementation using a dynamic programming approach is the nucleotide dependence in each pair of sequences, given by the substitution model. The triplet model is a 61 by 61 time-reversible codon substitution model described in the previous chapter 2. The approximation makes the nucleotides in each codon in the descendant sequence individually independent by calculating the probability that each ancestral codon produces specific descendant nucleotides at each reading frame position.

If we let $X = \{X_1X_2X_3\}$ and $Y = \{Y_1Y_2Y_3\}$ be codons from Σ_{codon}, composed by nucleotides $\{X_1, X_2, X_3\} \in \Sigma_{DNA}$ and $\{Y_1, Y_2, Y_3\} \in \Sigma_{DNA}$ respectively, the probability that the descendant codon Y substitutes the ancestral codon X after time $t \in \Theta$ by the triplet model is $P\left(Y_1Y_2Y_3|X_1X_2X_3;\Theta\right)$. Then, we define the approximate model substitution probability that the nucleotide X_p changes to nucleotide $y \in \Sigma_{DNA}$ using the plus semiring operation as

$$P_{\text{apx}}\left(Y_p = y|X_1X_2X_3;\theta\right) = \bigoplus_{Y_1Y_2Y_3} I(Y_p = y) \otimes P\left(Y_1Y_2Y_3|X_1X_2X_3;\Theta\right) \tag{3.1}$$

where θ contains the model parameters and is defined $\theta = \Theta \cup p$, p represents the position of the descendant nucleotide relative to the ancestral reading frame, defined $p \in \{1, 2, 3\}$, and $y \in \Sigma_{DNA}$ is the descendant nucleotide. Additionally, I is an indicator function that returns one if the left-hand side and right-hand side nucleotides are equal $I(e) = \{1 \text{ if } e \text{ is true and } 0 \text{ otherwise}\}$. The resulting model iterates over all non-stop 61 codons, for each of three reading frame positions in a codon and all four nucleotides, thus ending with a matrix of dimensions 61×3×4 stored as a 183×4 two-dimensional matrix (collapsing codon and reading frame position).

In the context of sequence alignment, the plus semiring operator ($\oplus$) computes the weight of a sequence and is the fitting operator to calculate the approximate probabilities. The semirings implemented in openFST and COATi define two plus operators sum (+) and max, which offer two different approaches to approximate substitution probabilities and results in two distinct models. The marginal model uses the linear semiring, with sum as its plus operator, and sums over all possible substitutions. This model was introduced in the previous chapter (Equ. 2.5). Secondly, the modal model is defined using the tropical semiring, with max as the plus operator. Conceptually, this model selects the best substitution over all possible codons. Note that the substitution probabilities are log-transformed after the marginalization. Otherwise, the marginal model would operate in log space using the log semiring ($\oplus_{log}$). In the subsequent sections, I analyze the fidelity of the marginal and modal models to the triplet model.

In statistical alignment, a common practice is to scale insertion probabilities according to the specific residue being added. While insertions in the triplet model are proportional to the nucleotide stationary frequencies, the modal and marginal models weigh insertions differently, subtracting the corresponding nucleotide stationary frequency for every residue in the descendant sequence. Consequently, the insertion probabilities become independent of the added residue, simplifying the alignment algorithm. It's important to note that while this scaling impacts the final alignment weight, it does not affect the relative weights between alignments of the same pair of sequences. This is because the subtracted probabilities only depend on the nucleotides in the descendant sequence, which remain constant for all alignments involving the same sequence pair, and thus the alignment procedure remains unaffected.

Mathematically, let s and v be an ancestral and a descendant sequence, respectively, with symbols in Σ_{DNA}. Consider A and B with entries in $\Sigma_{DNA} \cup \{-\}$ as

two distinct pairwise alignments of sequences s and v, Π_v as the product of stationary nucleotide frequencies in v, and $f()$ as the joint probability of the sequences and the alignment. Adding the scaling factor Π_v to the function $f()$ results in function $g()$. This function calculates the scaled joint probability of the sequences and the alignment, with the benefit that the nucleotides in v identified as insertions by the alignment are no longer dependent on their nucleotide frequency since this is canceled out by Π_v. While this changes the resulting weight of the alignments, the ratio of the two alignment probabilities remains unchanged, therefore the search for the best alignment is unaffected (as shown in Eq. 3.2).

$$\frac{g(s,v,A)}{g(s,v,B)} = \frac{f(s,v,A)\cdot(\Pi_v)^{-1}}{f(s,v,B)\cdot(\Pi_v)^{-1}} = \frac{f(s,v,A)}{f(s,v,B)} \tag{3.2}$$

After correcting for nucleotide stationary frequencies, the updated marginal and modal model definitions in linear space are

$$P'_{\text{marg}}\left(Y_p = y|X;\theta\right) = \sum_{Y} I(Y_p = y)\cdot P\left(Y|X;\Theta\right)\cdot(\pi_y)^{-1} \tag{3.3}$$

$$P'_{\text{modal}}\left(Y_p = y|X;\theta\right) = \max_{Y} \;\; I(Y_p = y)\cdot P\left(Y|X;\Theta\right)\cdot(\pi_y)^{-1} \tag{3.4}$$

where $X, Y \in \Sigma_{codon} - \{\text{TAA, TAG, TGA}\}$, Y_p and $y \in \Sigma_{DNA}$, $p \in \{1,2,3\}$. This updates equation 3.1, following its notation. Note that codons X and Y in equations 3.3 and 3.4 use a simplified notation that omits their constituting nucleotides.

3.2.4 Marginal and Modal Alignment Algorithm

The alignment algorithm for the approximate models in COATi is an adaptation to the Gotoh dynamic programming algorithm (Gotoh, 1982) described in the introduction chapter 1. Although my version of this renowned algorithm retains the

general structure and the three matrices match (M), deletion (D), and insertion (I), it increments the number of transitions to consider. Let s, v with symbols in Σ_{DNA} be two sequences with m and n residues, for each cell in the match matrix the Gotoh alignment considers:

$$D[i,j] = max \begin{cases} M[i-1,j] \otimes \alpha : \text{deletion opening} \\ D[i-1,j] \otimes \beta : \text{deletion extension} \end{cases}$$

$$I[i,j] = max \begin{cases} M[i,j-1] \otimes \alpha : \text{insertion opening} \\ I[i,j-1] \otimes \beta : \text{insertion extension} \end{cases}$$

$$M[i,j] = max \begin{cases} M[i-1,j-1] \otimes \text{score}(s_i, v_j) : \text{substitution} \\ D[i,j] : \text{deletion} \\ I[i,j] : \text{insertion} \end{cases}$$

where α and β are the gap opening and gap extension parameters, and `score()` is a function that scores the match or mismatch of two residues in Σ_{DNA}. Thus, using the tropical semiring and the model parameters defined in the previous subsection, each cell in the dynamic programming alignment algorithm matrices satisfies one of the following recursions:

$$D[i,j] = \oplus \begin{cases} M[i-1,j] \otimes \log_{1m}(g) : \text{deletion opening after match} \\ D[i-1,j] \otimes e : \text{deletion extension} \\ I[i-1,j] \otimes \log_{1m}(e) \otimes g : \text{deletion opening after insertion} \end{cases}$$

$$I[i,j] = \oplus \begin{cases} M[i,j-1] \otimes g : \text{insertion opening after match} \\ I[i,j-1] \otimes e : \text{insertion extension} \end{cases}$$

$$M[i,j] = \oplus \begin{cases} M[i-1,j-1] \otimes \log_{1m}(g) \otimes (1-g) \otimes P'_{mar}[s_i, v_j; \theta] : \text{substitution} \\ D[i-1,j-1] \otimes \log_{1m}(e) \otimes P'_{mar}[s_i, v_j] : \text{substitution} \\ I[i-1,j-1] \otimes \log_{1m}(e) \otimes \log_{1m}(g) \otimes P'_{apx}[s_i, v_j] : \text{substitution} \end{cases}$$

where the function $\log_{1m}(x)$ performs the operation $\log(1 - exp(-x))$ and is used to calculate the probability of not opening a gap $log_{1m}(g)$ or not extending a gap $log_{1m}(e)$. This function is implemented following (Mächler, 2012).

While the Gotoh algorithm stores the best transition for each cell in the match matrix, COATi does not. The backtracking algorithm finds the best end value across the three matrices and sets it as the starting point to retrieve the optimal alignment. Note that the insertion matrix I does not consider a transition from the deletion matrix D, as designed in the indel approximate model (Fig 3.2). In addition, the last cell on all matrices ($M_{m,n}$, $D_{m,n}$, $I_{m,n}$) is updated with the end transition weights (Eq. 3.5), albeit is not shown in algorithm 3 for simplicity.

$$
\begin{aligned}
D_{m,n} &= D_{m,n} \otimes \log_{1m}(e) \\
I_{m,n} &= I_{m,n} \otimes \log_{1m}(e) \otimes \log_{1m}(g) \\
M_{m,n} &= M_{m,n} \otimes \log_{1m}(g) \otimes \log_{1m}(g)
\end{aligned}
\tag{3.5}
$$

Furthermore, a feature of the algorithm is the ability to restrict gap unit length. This allows users to impose specific gap lengths, such as only allowing gaps to be in three-mers, maintaining the reading frame. An example of this can be seen in the evolutionary analyses of indel rates by Zhu (2022), where protein-coding sequences were aligned with COATi restricting gaps to be lengths which are multiples of three (i.e., no frameshifts). Fixing gap unit length is not shown in algorithm 3, which assumes the general case where gaps can be of any length (unit of 1).

3.2.5 Sequence Processing and Encoding

COATi pre-processes the input sequences before alignment to ensure certain conditions are met and to allow the use of speed-up techniques. One of the assumptions is that the ancestor protein-coding sequence is of high quality and is used to help preserve the reading frame and safeguard against frameshifts in the potentially low-quality descendant sequence. Therefore, the ancestor sequence must be of length multiple of three (no incomplete codons) and be free of any ambiguous nucleotides or early stop codons. In contrast, the descendant sequence can be of any length, contain nucleotide codes included in IUPAC (Cornish-Bowden, 1985), and include early stop codons.

After the initial validations, COATi encodes the sequences as vectors of `unsigned char`, an efficient C++ eight-bit character representation that can be used to access

Algorithm 3 Marginal pairwise alignment algorithm. Intentionally different from the Gotoh algorithm to implement the marginal evolutionary model in COATi.

function ALIGNPAIR(s, v, g, e)
  $n, m \leftarrow |s|, |v|$
  M, D, I $\leftarrow$ zero$(n+1, m+1)$ ▷ Initialize the matrices of size $n+1$ by $m+1$
  for $i \leftarrow 1$ to n **do** ▷ Gap deletion penalties
    $\mathrm{D}[i, 0] \leftarrow g \oplus ((i-1) \otimes e)$
  end for
  for $j \leftarrow 1$ to m **do** ▷ Gap insertion penalties
    $\mathrm{I}[0, j] \leftarrow g \oplus ((j-1) \otimes e)$
  end for
  $M[0, 0] \leftarrow \overline{1}$ ▷ Set starting value in match matrix
  for $i \leftarrow 1$ to n **do** ▷ Fill in the matrices
    for $j \leftarrow 1$ to m **do**
      mch2mch $\leftarrow M[i-1, j-1] \otimes \log_{1m}(g) \otimes \log_{1m}(g) \otimes \mathrm{P}_{apx}[s_i, v_j]$
      del2mch $\leftarrow D[i-1, j-1] \otimes \log_{1m}(e) \otimes \mathrm{P}_{apx}[s_i, v_j]$
      ins2mch $\leftarrow I[i-1, j-1] \otimes \log_{1m}(e) \otimes \log_{1m}(g) \otimes \mathrm{P}_{apx}[s_i, v_j]$
      mch2del $\leftarrow M[i-1, j] \otimes \log_{1m}(g) \otimes g$
      del2del $\leftarrow D[i-1, j] \otimes e$
      ins2del $\leftarrow I[i-1, j] \otimes \log_{1m}(e) \otimes g$
      mch2ins $\leftarrow M[i, j-1] \otimes g$
      ins2ins $\leftarrow I[i, j-1] \otimes e$
      $M[i, j] =$ mch2mch $\oplus$ del2mch $\oplus$ ins2mch ▷ Save match scores
      $D[i, j] =$ mch2del $\oplus$ del2del $\oplus$ ins2del ▷ Save deletion scores
      $I[i, j] =$ mch2ins $\oplus$ ins2ins ▷ Save insertion scores
    end for
  end for
  Add end weights
  Traceback
end function

matrices. This procedure uses a lookup table called `nt16_table` that maps each nucleotide (A/a, C/c, G/g, T/t/U/u) to a corresponding value (0, 1, 2, 3). This enables an efficient conversion of codons from strings to `unsigned char` using bit-wise operators where each nucleotide takes two of the eight bits (two left-most bits are unused), and each codon is converted to a corresponding value from 0 (AAA) to 63 (TTT), using the following C++ algorithm:

```
int cod_int(const std::string_view codon) {
    return (nt16_table[codon[0]] << 4) |
           (nt16_table[codon[1]] << 2) |
            nt16_table[codon[2];
}
```

The marginal and modal substitution models are stored as a 183×4 matrix, where each row represents an ancestral codon and a reading frame position (phase), and each column represents a descendant nucleotide. Thus, the encoded sequences must contain codon, nucleotide, and phase information to access a value in such a matrix. The descendant sequence is converted to an encoded vector of nucleotides, while the ancestor encoded sequence contains codons and phases. While creating the vector of nucleotides is straightforward, the ancestor sequence combines codons and position as shown in table 3.2. Note the codon table used here does not contain stop codons (TAA/UAA, TAG/UAG, TGA/UGA).

Encoding the input sequences allows the core alignment algorithm to use the values to access the substitution probabilities directly. Since matrix indexing is one of the most used operations, this helps reduce runtime costs. In addition, the encoding process can be conveniently reverted to retrieve a nucleotide from an encoded codon, given its position. This operation is included as the function `get_nuc()`, which also

Codon*	Encoded Codon	Position	Encoded Codon + Position
string	unsigned char	int	codon · 3 + position
AAA	0	0	0
AAA	0	1	1
AAA	0	2	2
AAC	1	0	3
AAC	1	1	4
AAC	1	2	5
⋮	⋮	⋮	⋮
TTT	60	0	180
TTT	60	1	181
TTT	60	2	182

Table 3.2: Encoding of codon and position. Each codon is converted to a corresponding value [0, 60], multiplied by 3, and then added a position offset. This results in assigning each codon and position a value [0, 182]. *Stop codons are not included.

takes advantage of bit operators (Table 3.3), and is used to calculate the marginal and modal substitution probabilities (Alg. 4).

Codon	uchar	Position	Mask	uchar & mask ≫ shift = Nucleotide
ACG	00 **00** **01** **10**	0	00 **11** 00 00	00 **00** 00 00 ≫ 4 = **00** (**A**)
ACG	00 **00** **01** **10**	1	00 00 **11** 00	00 00 **01** 00 ≫ 2 = **01** (**C**)
ACG	00 **00** **01** **10**	2	00 00 00 **11**	00 00 00 **10** ≫ 0 = **10** (**G**)

Table 3.3: Extraction of nucleotides from an encoded codon using bit-wise operators. Each nucleotide takes two bits that are masked and shifted according to the position specified.

Algorithm 4 Marginal substitution algorithm. Substitution probabilities for the triplet model are in P_{tri}, and π contains nucleotide frequencies.

function MARGINAL-SUBSTITUTION(P_{tri}, π)
    Create the substitution matrix of size 183x4
    $P_{apx} \leftarrow \text{zero}(n+1, m+1)$
    for cod $\leftarrow$ 1 to 61 **do**
        for nuc $\leftarrow$ 1 to 4 **do**
            for pos $\leftarrow$ 1 to 3 **do**
                tmp $\leftarrow$ 0
                for i $\leftarrow$ 1 to 61 **do**
                    if get_nuc(i, pos) == nuc **then**
                        tmp $\otimes =$ P_{tri}[cod, i]
                    end if
                end for
                P_{apx}[cod $\cdot$3+pos, nuc] $= \log\left(\frac{\text{tmp}}{\pi_{nuc}}\right)$
            end for
        end for
    end for
    return P_{apx}
end function

3.3 Methods

3.3.1 Benchmark Dataset Simulation

Simulating sequence evolution plays an essential role in bioinformatics as an indispensable tool for validating novel methods, evaluating the performance of phylogenetic methods, and testing hypotheses, among other techniques (Ly-Trong *et al.*, 2022). In sequence alignment, benchmark datasets are frequently used to assess alignment algorithms and estimate model parameters under diverse evolutionary conditions. When curated datasets are unavailable or unsuitable for specific validation requirements, it is common practice to generate them using simulators. Several DNA sequence alignment simulators, including DAWG (Cartwright, 2005), INDELible (Fletcher and Yang, 2009), and AliSim (Ly-Trong *et al.*, 2022), have been devel-

oped to replicate evolutionary processes in a range of parameter-rich models. Nevertheless, empirical simulations often provide a more accurate assessment, as they mimic natural evolution more accurately.

Therefore, I developed a testing pipeline that features an empirical evolution simulation algorithm to evaluate how well the approximate models approximate the triplet model. First, I downloaded 16,000 human protein-coding genes (CDS) from the ENSEMBL database (Hubbard *et al.*, 2002). After downloading, I filtered out 2,226 sequences longer than 3,000 nucleotides to limit runtime and memory costs. I discarded an additional 16 CDS that contained early stops or incomplete codons (i.e., their length was not multiple of three). Subsequently, I generated a set of five distinct datasets of pairwise sequence alignments, each dataset using each with a different branch length of 0.2, 0.4, 0.6, 0.8, or 1.0, to test the models under different evolutionary distances. Every dataset was created using a simulation algorithm that follows the triplet model. I used these datasets to evaluate the accuracy of the triplet, marginal, and modal models. After removing the gaps, I aligned the sequences using all three methods and measured their performance. I ran this pipeline twice, one for each substitution model available in COATi (i.e., MG94 and ECM). Additional information, code, and workflows to replicate the analysis can be found on GitHub: `https://github.com/jgarciamesa/coati-evo-sim`.

The evolution simulation algorithm takes an input CDS, a strength of selection coefficient ω, and a branch length t (number of expected substitutions per site) to generate two descendant sequences. Initially, I introduce substitutions using the triplet substitution model with branch length $t/2$, default ω, gap open, and gap extension parameters. Then, I simulate gaps along the sequence following the geometric indel model. Notably, when working with CDS, gap lengths not multiple of three, known as frameshifts, are considered artifacts that disrupt the open reading frame and are

likely to be purged by purifying selection. As a result, the simulation algorithm does not model the introduction of errors and only considers non-frameshift gaps. This is duplicated to generate a second descendant sequence and form a pairwise alignment with distance t. The process is then repeated for every one of the 13,758 filtered sequences to create a benchmark dataset.

3.3.2 Alignment Metrics

Sequence alignment methods are often evaluated by their capacity to retrieve alignments from a benchmark dataset (e.g., Sievers *et al.* (2011)). This assessment involves the utilization of one or more distance metrics to gauge how closely the output generated by the aligners matches the reference dataset. Commonly used scoring methods include the sum-of-pairs score (SP), which quantifies the proportion of correctly identified residue pairs, and the total column score (TC), which calculates the fractions of reference columns found. However, SP and TC are not actual metrics since symmetry is not guaranteed when calculating the alignment distance between two alignments (i.e., $\mathrm{SP}(A \to B) \neq \mathrm{SP}(B \to A)$ for two given alignments A and B) (Blackburne and Whelan, 2011).

To quantify the similarity between each alignment in the benchmark datasets and the corresponding output obtained from the triplet and marginal models, I used the alignment error metric d_{seq} (Blackburne and Whelan, 2011). This metric accounts for indels and is more informative than conventional distance measures like SP or TC. Intuitively, d_{seq} ranges between zero and one and can be interpreted as the probability that a randomly selected residue will be aligned to a different location against a sequence that does not contain such residue.

The computation of d_{seq} involves characterizing the gaps present in the alignment. Then, a site-wise homology set $H(A)^i_j$ is calculated for each alignment A, sequence i,

and character j. The distance between two alignments A and B is the average across all characters of the symmetric difference (or Hamming distance), represented as '$\triangle$' between homology sets over the length of such sets:

$$d(A,B) = \frac{1}{c}\sum_{i}\sum_{j}\frac{|H(A)^i_j \triangle H(B)^i_j|}{|H(A)^i_j| + |H(B)^i_j|} \tag{3.6}$$

where c is the sum of the sequence lengths.

To quantify the results further, I compared the number of perfectly and imperfectly retrieved alignments for each model. Perfect alignments are defined as those with a distance of zero to the reference alignment ($d_{seq} = 0$), indicating 100% similarity. Notably, a set of sequences can have more than one optimal alignment under the same evolutionary model, despite algorithms typically producing a single result. Consequently, to account for evolutionary equivalent alignments, I scored all alignments using the marginal-mg model and considered those with scores identical to the benchmark alignments as perfect. Furthermore, I computed the count of imperfect alignments where an alignment is considered imperfect when its distance to the reference alignment is greater than zero ($d_{seq} > 0$) and another method successfully produces an alignment with 100% similarity. This analysis exposes instances where all models fall short of achieving a perfect result in addition to a direct model-to-model comparison.

3.3.3 Kullback-Leibler Divergence

The substitution probabilities in the triplet model are stored in a 61 by 61 matrix. Each cell within this matrix represents the probability of replacing a codon, given by the column, with another, given by the row. In turn, the approximate models decompose these 61 probabilities for each column into 12 segments, each representing the

replacement of the codon with one of four possible nucleotides in one of three positions within a codon. To compare how well their substitution probabilities approximate the triplet model, I reconstructed the 61 by 61 codon-to-codon probabilities from the approximate models. While this framework transforms the mutation rates, it allows a direct comparison. If we let $X, Y \in \Sigma_{codon}$ and P_{apx} be the substitution probabilities using the marginal or modal model, the probability that codon Y substitutes codon X after time t is calculated

$$P_{apx}(Y_1|X;\theta) \cdot P_{apx}(Y_2|X;\theta) \cdot P_{apx}(Y_3|X;\theta)$$

where $Y_p \in \Sigma_{DNA}$ represents the nucleotide in position p of codon Y, and the calculations are performed in linear space.

I used the Kullback-Leibler divergence (D_{KL}) to assess how well the marginal and modal models approximate the triplet model as a whole under different branch lengths. The D_{KL} measures how one probability distribution differs from a second reference probability distribution, introduced by Kullback and Leibler in information theory (1951). Namely, it quantifies the amount of information lost when one distribution is used to approximate another. Mathematically, if you have two discrete probability distributions $P(x)$ and $Q(x)$ defined on the same sample space χ, the Kullback-Leiber divergence from Q to P is defined

$$D_{KL}(P|Q) = \sum_{x \in \chi} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{3.7}$$

where x represents individual events or values, $P(x)$ is the probability of event x in the distribution P, and $Q(x)$ is the probability of event x in the distribution Q. Therefore, I used D_{KL} to measure how well the approximate models estimate the triplet model,

where $Q = P_{apx} \cdot \pi$, $P = P_{triplet} \cdot \pi$, π are the codon stationary frequencies, and P_{apx} is the codon-to-codon substitutions obtain from the approximate models. The calculations were done in the statistical programming language R using the function `KLD` from the LaplacesDemon package (Statisticat and LLC., 2021).

3.3.4 Alignment Visualization

A common workflow in evaluating methods of alignment inference is to use a distance score to quantify the accuracy of the results, such as SP, TC, or in my case, d_{seq}. While these statistics are informative, they often fail when trying to identify subtle differences between models. To address this issue, I designed and developed a program (AlnDotPlot) that detects variations between two alignments, described in chapter 4. This tool can take two alignments from the same pair of sequences as input, find the section where they diverge, and generate a visual representation of such segment, as demonstrated in Figure 3.3. These visualizations, hereby alignment dot plots or dot plots, are presented as two-dimensional grids, where each row corresponds to nucleotides in the first sequence, and each column represents nucleotides in the second sequence. I introduce padding columns and rows marked with '-' to account for indels as needed. Nodes within the grid correspond to matches, mismatches, or gaps defined by the alignment, and edges connect them to form a path. Each alignment is assigned a unique color, except when the paths merge, creating a unique black line that indicates a shared alignment section. In addition, matching residues in the alignment are marked with an 'X' in the corresponding matrix cell.

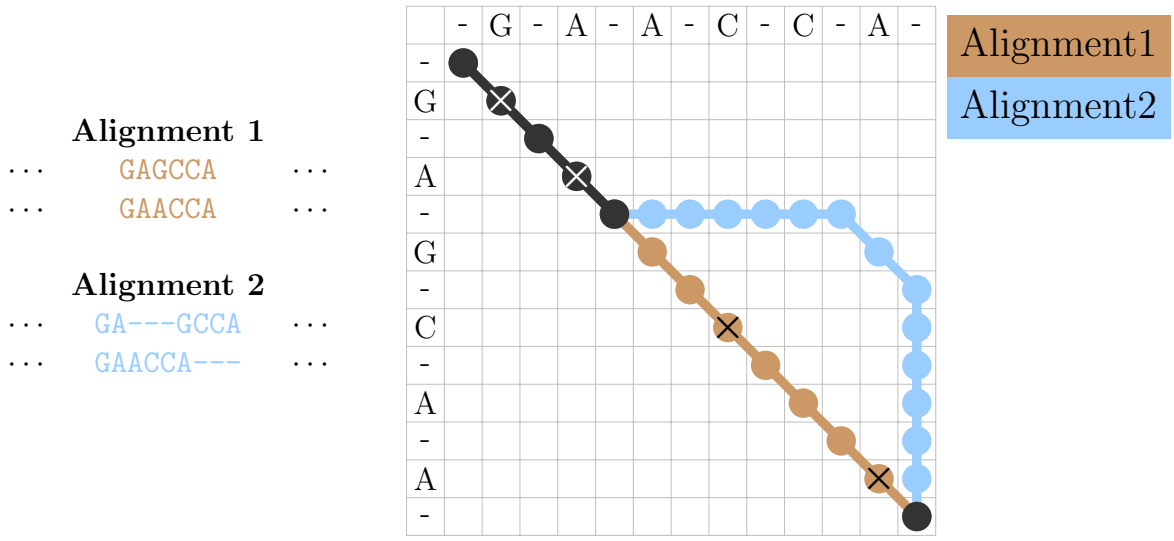


Figure 3.3: Example of a visual representation of the region where two pairwise alignments differ. Both sequences are 180 nucleotides long, although only the section where they differ is shown. The figure displays two preceding residues to show complete codons. Alignment 1 is colored in brown and alignment 2 is colored in blue, while matching portions of the alignment are in black. Nodes represent matches, mismatches, and gaps, while edges connect them to form a path. Matches in the alignment are marked with an 'X' at the corresponding node.

3.4 Results

3.4.1 Alignment Accuracy

MG94

The triplet MG94 model consistently outperformed the marginal MG94 models across all datasets (Fig. 3.4). At the lowest branch length of 0.2, the average alignment error among all three models is similar. However, an evident order emerges, with the triplet-mg model outperforming marginal-mg, which, in turn, achieved a smaller d_{seq} than modal-mg. As the number of expected substitutions increases, the d_{dseq} for the marginal-mg model consistently remains close to that of the triplet model, although the latter outperforms the former across all branch lengths. In contrast, the average alignment error for the modal-mg model diverges further from the other models, spiking when, on average, we expect every site to have undergone a mutation (branch length of 1), with a d_{seq} more than twofold its previous value.

The number of perfect alignments between the marginal-mg and triplet-mg model remains consistent across all branch lengths. In contrast, the marginal-mg starts producing an equal number of perfect alignments for a branch length of 0.2, but this number declines as branch lengths increase. A similar trend is observed in the count of imperfect alignments, with the modal-mg model producing more imperfect alignments, particularly spiking at branch lengths of 0.8 and 1.0. In this section, the remaining models performed similarly, with the marginal-mg model outperforming the triplet-mg model with branch lengths 0.6 and 0.8, with the reverse outcome for branch lengths 0.2, 0.4, and 1.0. Note that the count of perfect and imperfect alignments decreases along the x-axis, as alignments not perfectly retrieved by either method are excluded from the results.

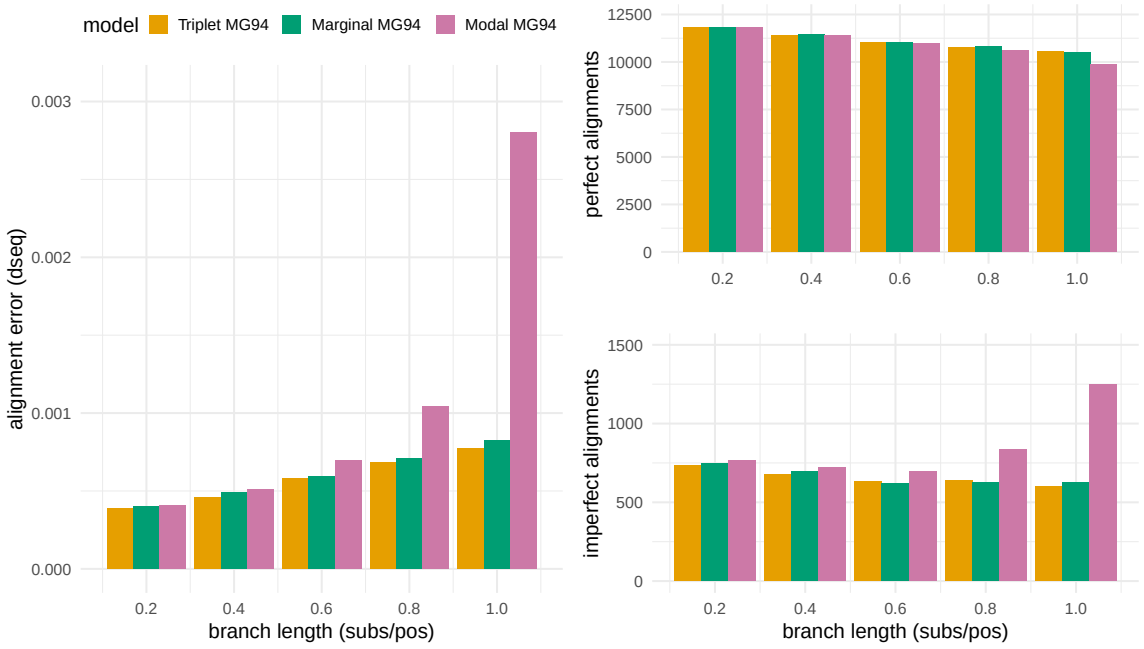


Figure 3.4: The triplet-mg model generates better alignments across all branch lengths. Results of triplet-mg, marginal-mg, and modal-mg COATi models in aligning 13,758 simulated sequence pairs. Best alignments have the lowest d_{seq}, perfect alignments have the same score as the true alignment or a zero d_{seq}, and imperfect alignments have a different score than true alignments when at least one model found a perfect alignment.

ECM

As expected, the triplet-ecm model outperforms the marginal models across all branch lengths in all metrics (Fig. 3.5). The trend observed in the MG94 results is intensified in the ECM results. The d_{seq} values for the triplet-ecm and marginal-ecm models are comparable to their MG94 counterparts, albeit with a slightly larger difference between them. This pattern is also evident in the number of perfect and imperfect alignments, where the triplet-ecm model significantly outperforms the approximate models. However, the marginal-ecm model provides a better approximation. In contrast, the modal-ecm model underperforms with a d_{seq} two orders of magnitude larger than its MG94 counterpart for a branch length of 1. The model struggles

with short branch lengths, and its performance declines as the number of expected substitutions per site increases. I have truncated the d_{seq} values at 0.05 to ensure a proper display of the results for the triplet-ecm and marginal-ecm. A comprehensive results table can be found in the appendix (Table B.1).

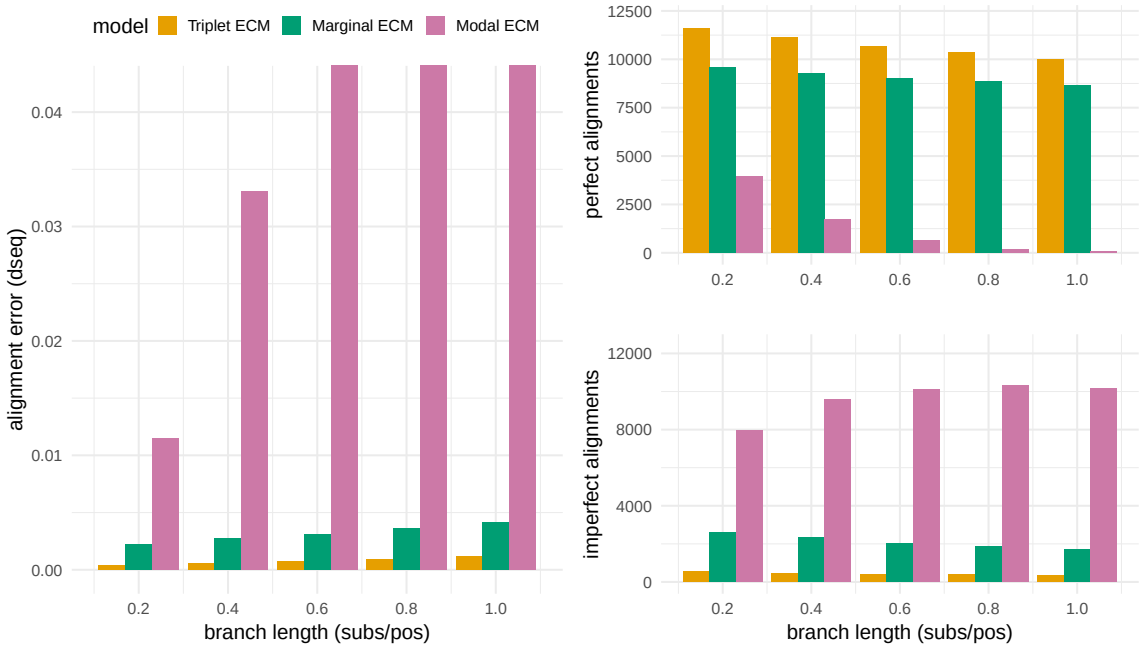


Figure 3.5: The triplet-ecm model generates better alignments across all branch lengths. Results of triplet-ecm, modal-ecm, and marginal-ecm COATi models in aligning 13,758 simulated sequence pairs. Best alignments have the lowest d_{seq}, perfect alignments have the same score as the true alignment or a zero d_{seq}, and imperfect alignments have a different score than true alignments when at least one model found a perfect alignment.

3.4.2 Gap Statistics

Gap statistics can provide valuable insights when comparing alignment models across varying evolutionary distances, as they reveal how the likelihood of substitution relative to indel probabilities changes. Notably, for both the MG94 and ECM models, the total number of gaps and their cumulative length remains constant for both the

triplet and marginal models as branch lengths increase (Fig. 3.6). In concordance with the previous metrics, the behavior of the modal model takes a divergent path, especially for modal-ecm. While the modal-mg model shows similar values until branch length reaches 0.6 and is slightly elevated thereafter, the total number and length of gaps for the modal-ecm model are larger for all branch lengths.

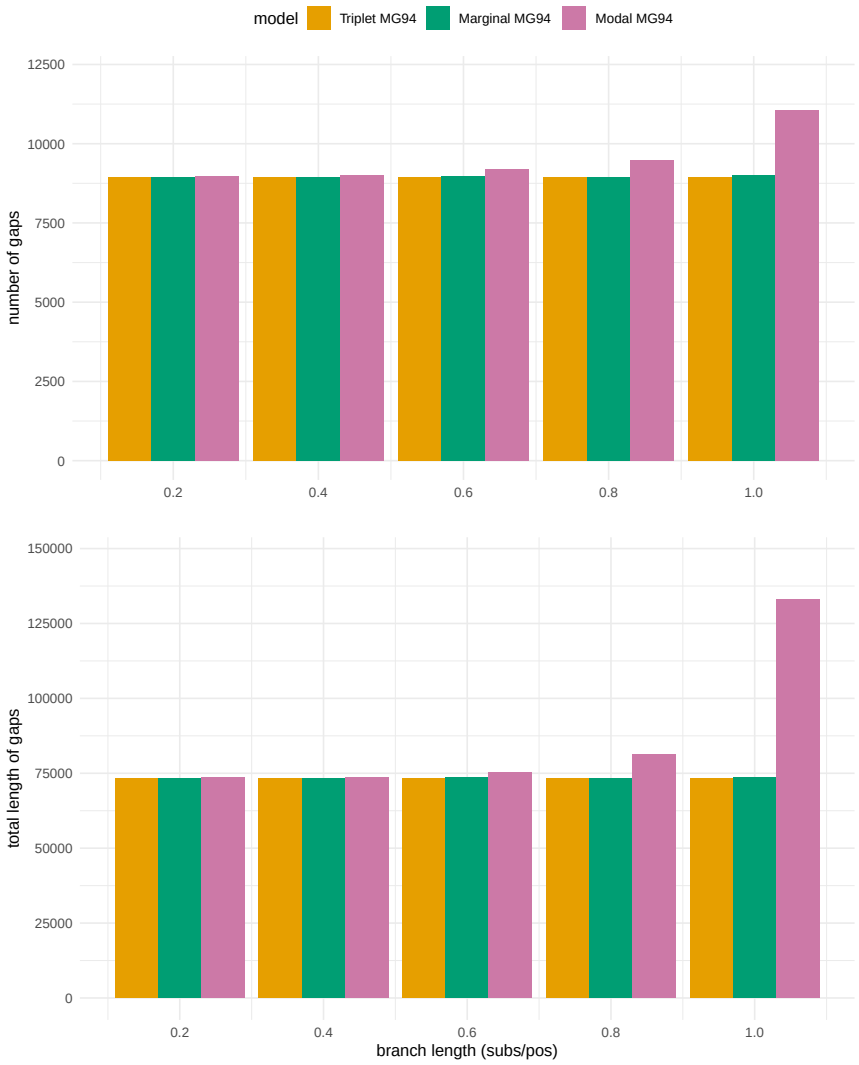


Figure 3.6: The number and total length of gaps for triplet-mg and marginal-mg models stay constant as branch lengths increase. On the contrary, the modal-mg model adds more and longer gaps as the evolutionary distance between sequences becomes larger.

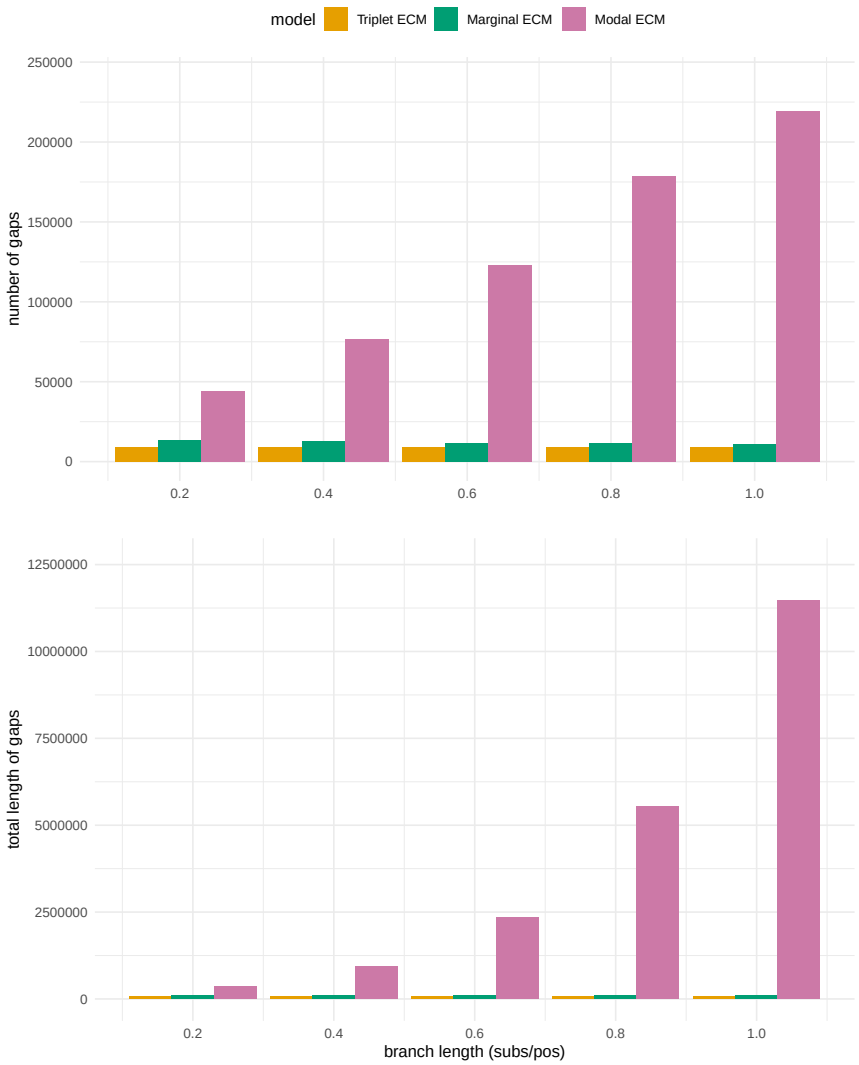


Figure 3.7: The number and total length of gaps for triplet-ecm and marginal-ecm models stay constant as branch lengths increase. On the contrary, the modal-ecm model significantly adds more and longer gaps as the evolutionary distance between sequences becomes larger.

3.4.3 Marginal Model

The results from the previous section establish the marginal model as a close approximation to the triplet model. To better understand how marginalization influences the substitution probabilities, I plotted the Kullback-Leibler divergence (D_{KL}) between the marginal and the triplet model. This score can be seen as the measure of how one probability distribution, the marginal, differs from a reference probability distribution, the triplet. Figure 3.8 is a plot of D_{KL} for the MG94 marginal (top) and ECM marginal (bottom) models with branch length values from 0.1 to 10. The divergence for both marginal models follows a similar trend, with values rising until they reach saturation to then tend towards zero. Marginal-mg is a better approximation to its triplet counterpart than marginal-ecm with smaller values across all branch lengths.

In addition to the overall measure of divergence, I plotted the D_{KL} matrix between the triplet and marginal model at a branch length of 1 to understand what substitutions drive this score. This 61 by 61 matrix plot represents individual divergence values calculated using D_{KL} (Eq. 3.7), where blue cells indicate substitution probabilities that are underestimated by the marginal model, while red cells indicate overestimation. Figure 3.9 shows the divergence scores for the marginal model. The divergence is driven by a combination of an underestimation of transversions and an overestimation of transitions on the first and third-position mutations. In this case, the most underestimated amino acids are Leucine and Serine, while the most overestimated are tryptophan and tyrosine.

The divergence matrix for the ECM model is plotted following the codon order used in the original ECM publication (Kosiol *et al.*, 2007), where the order of nucleotides is {T, C, A, G}, and first position changes precede second position ones

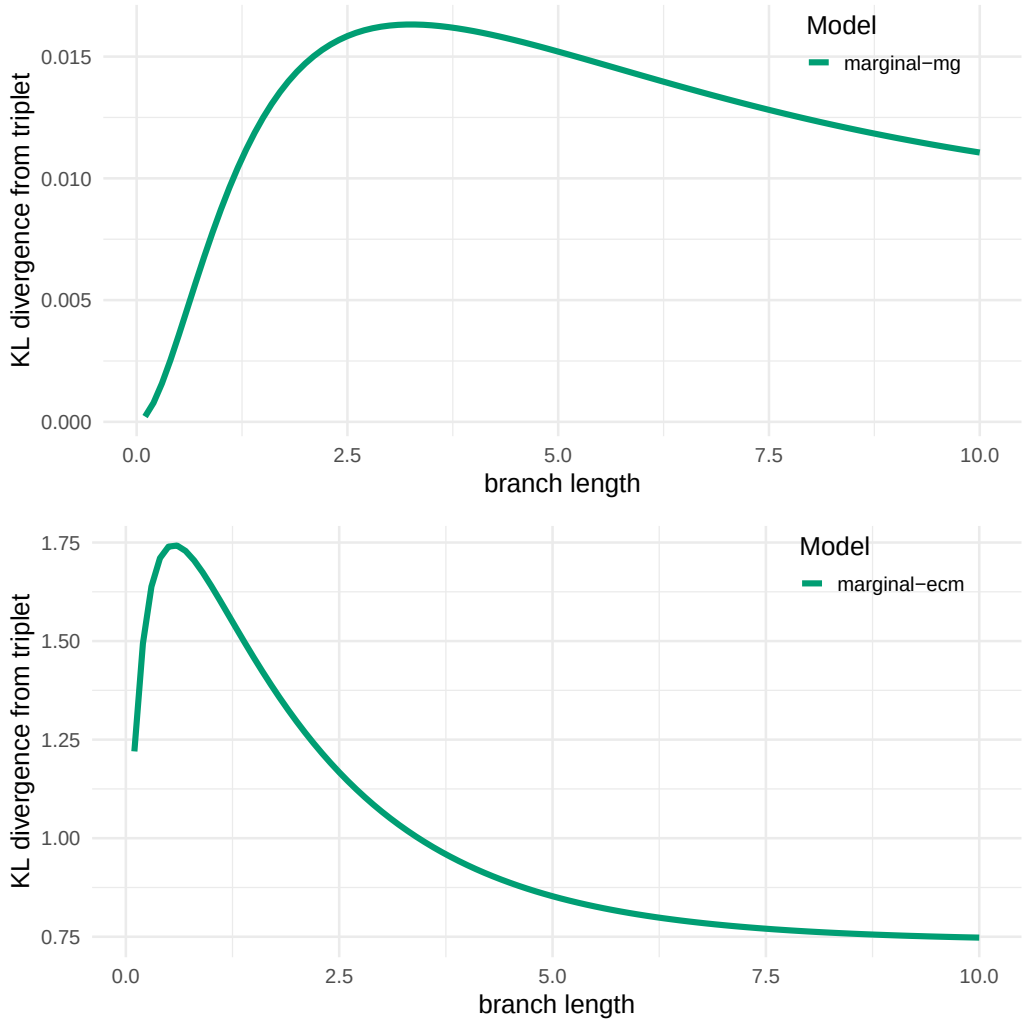


Figure 3.8: Kullback-Leibler divergence between MG94 (top) ECM (bottom) and their marginal models with branch lengths between 0.2 and 10. Low values represent a small divergence, indicating a better approximation to the triplet model. Marginal MG94 performs best, while the divergence for both models decreases as they reach saturation.

(Fig. 3.10). The D_{KL} divergence for the marginal-ecm model is driven by a general overestimation of substitution probabilities, especially along the main diagonal (indicating no change). In addition, we observe an underestimation of the arginine and serine amino acids (top left blue section). The most overestimated amino acids include histidine, although the difference is small. Notably, the divergence values for

the marginal ECM model are much higher than for the MG94 counterpart.

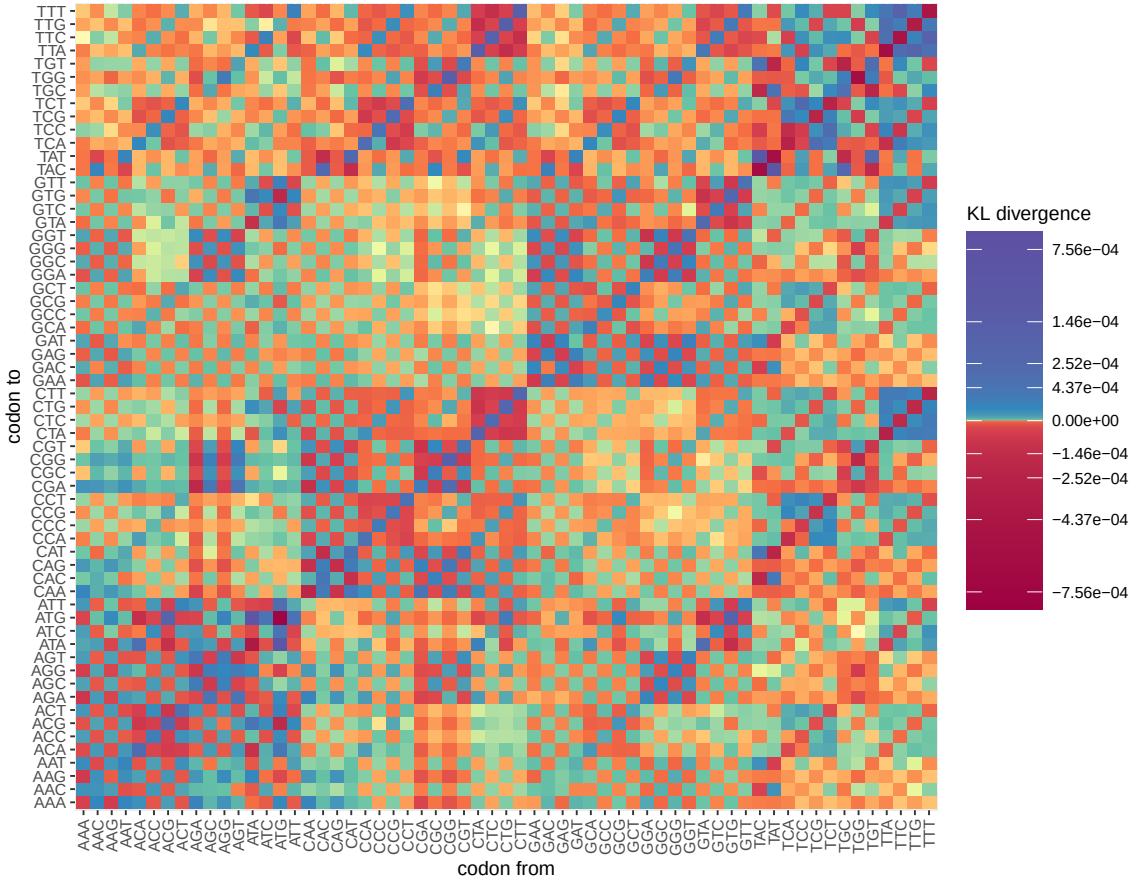


Figure 3.9: Kullback-Leibler divergence matrix for the marginal MG94 model with a branch length of 1.0. Values closer to zero indicate a smaller divergence, representing a better approximation to the triplet model. Positive values, indicated by a blue gradient, mark substitution probabilities where the marginal model underestimates the triplet model. In turn, negative values, indicated by a red gradient, represent substitution probabilities where marginal overestimates the triplet model.

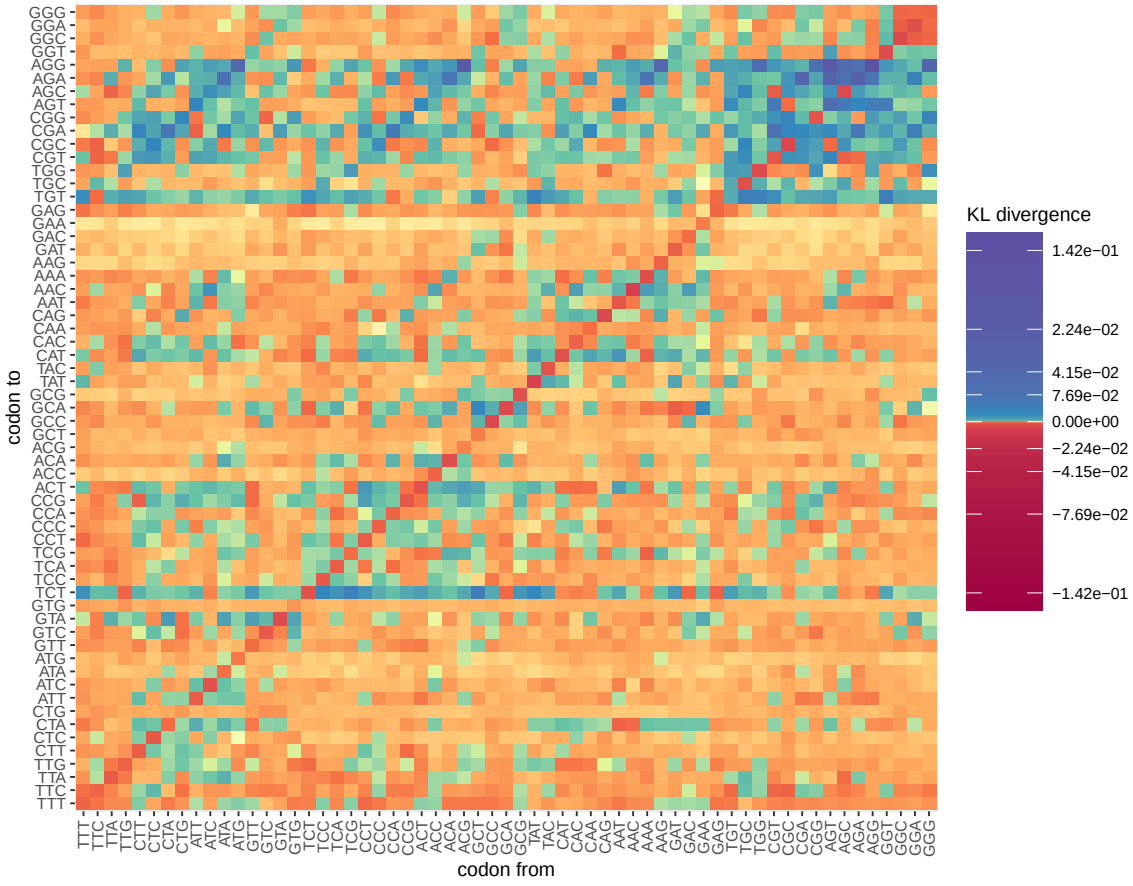


Figure 3.10: Kullback-Leibler divergence matrix for the marginal ECM model with a branch length of 1.0. Values closer to zero indicate a smaller divergence, representing a better approximation to the triplet model. Positive values, indicated by a blue gradient, mark substitution probabilities where the marginal model underestimates the triplet model. In turn, negative values, indicated by a red gradient, represent substitution probabilities where marginal overestimates the triplet model.

3.4.4 Visual Comparison of Triplet and Marginal Model

The previous metrics have demonstrated that the triplet and marginal models exhibit a strong agreement in their results, thereby validating the latter as a suitable approximation of the former. However, these metrics have fallen short in identifying any subtle distinctions between the models, if they exist. Pursuing these differences, I inspected the alignment sections where the two models diverge using dot plots. Upon visual comparison of the most distinct triplet and marginal alignments across sequence lengths spanning from a few hundred to a maximum of three thousand nucleotides and encompassing all branch lengths (0.2, 0.4, 0.6, 0.8, and 1.0), one predominant pattern emerges (examples in Fig. 3.11). The most notable distinction between the triplet and marginal models lies in the former model displaying a preference for substitutions, irrespective of the number of matches, whereas the latter model favors indels, often with a higher number of matches (left column in Figure 3.11).

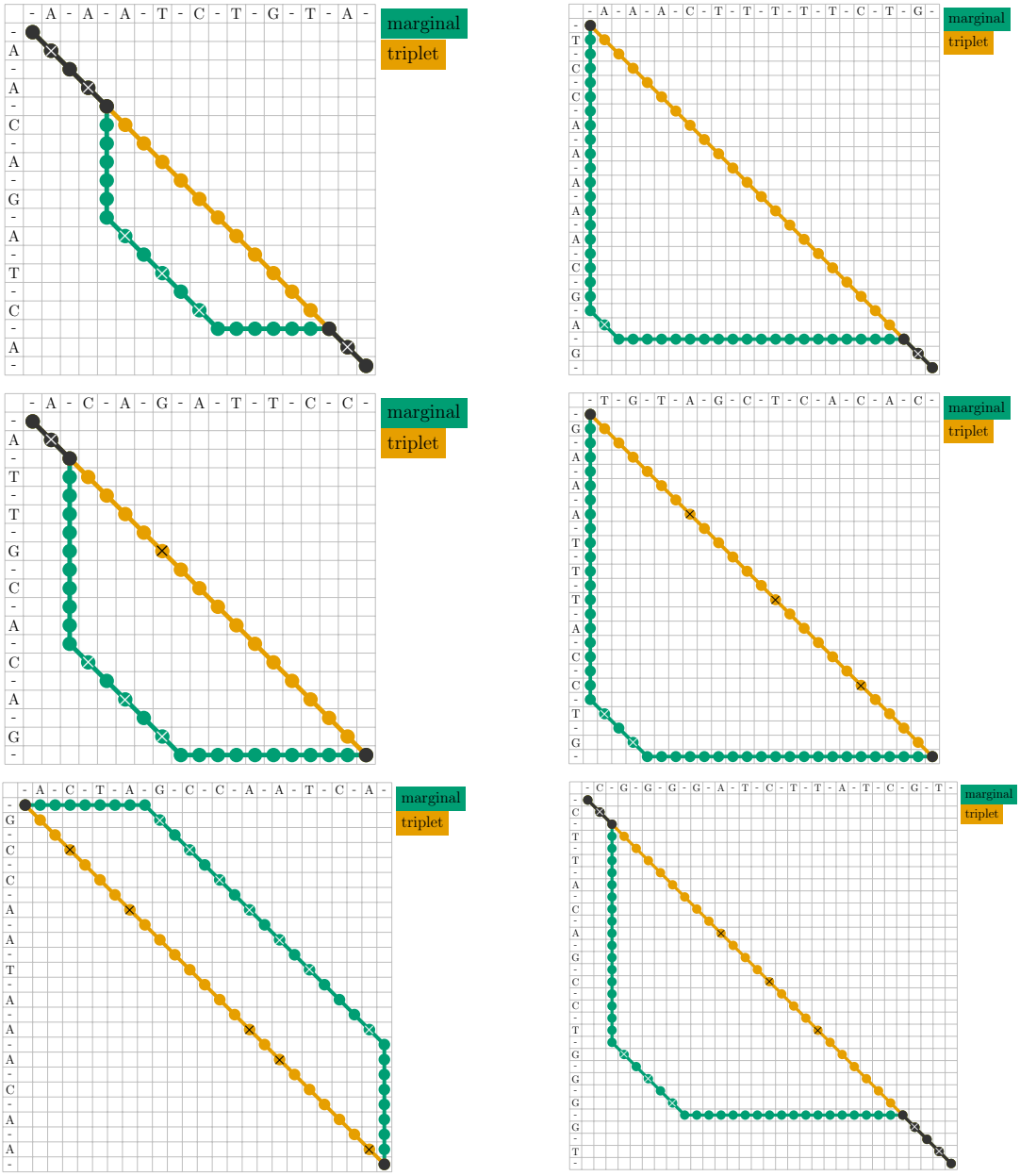


Figure 3.11: Dot plots of the sections where the triplet, marked in orange, and marginal, colored in green, model alignments differ. Matching nucleotides are marked with an 'X'. This selection of dot plots showcases the most common pattern in alignment diverging regions between the models, where the triplet model matches the residues. Instead, the marginal model finds its optimal path through indels and fewer substitutions. This trend is shared for the MG94 and the ECM codon substitution models. Therefore, this selection of dot plots is a combination of results using triplet-mg against marginal-mg and triplet-ecm against marginal-ecm.

3.4.5 *Runtime*

The motivation behind developing the marginal and modal models is to speed up sequence alignment in COATi. To showcase the improvement, I measured the speed of the triplet and marginal models together with a suite of popular aligners spanning various alignment methods: amino-acid based ClustalΩ v1.2.4 (Sievers *et al.*, 2011), amino-acid plus frameshifts MACSE v2.06 (Ranwez *et al.*, 2011), DNA version of MAFFT v7.505 (Katoh and Standley, 2013), and codon version of PRANK v.150803 (Löytynoja, 2014). A comprehensive comparison of the accuracy of these models against COATi can be found in chapter 2.

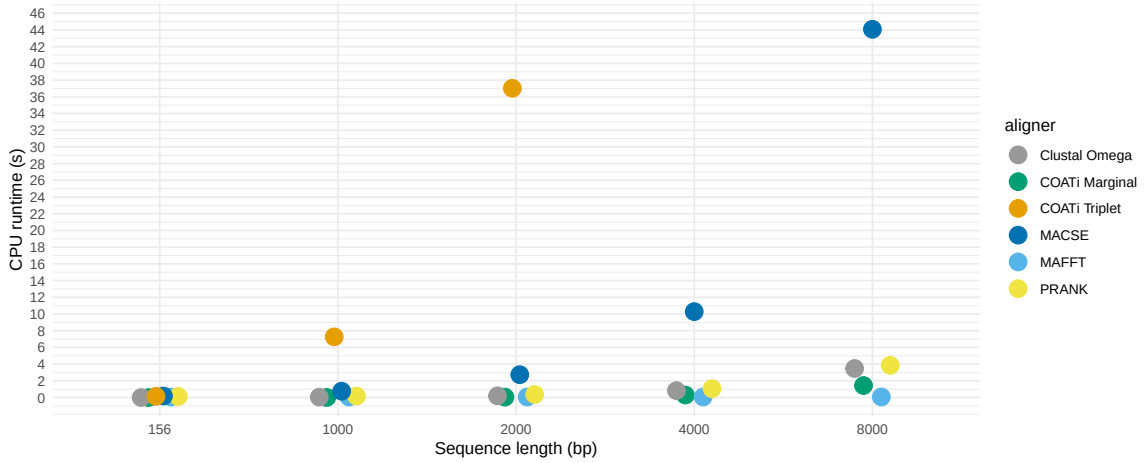


Figure 3.12: Execution time of benchmark in seconds of ClustalΩ, COATi triplet and marginal model, MACSE, MAFFT, and PRANK aligning pairwise sequences of different lengths. COATi triplet, implemented using FSTs, suffers from a costly runtime compared to other aligners. In comparison, COATi marginal solves the issue and can perform similarly to Clustal Ω, MAFFT, and PRANK and better than MACSE. This was run on an 11^{th} generation Intel chip with a single core.

The results (Fig. 3.12) show the execution time of the aligners with different sequence lengths. The runtime for the triplet model (orange) rapidly grows and can become a limitation when sequences exceed a thousand base pairs. Notably, the COATi marginal (green) is comparable to popular tools and considerably faster than

MACSE for longer sequence pairs. Both modal and marginal models are reported under COATi marginal because, despite their different definitions, their alignment algorithms are identical, therefore having matching execution times.

3.5 Discussion

While the statistical pairwise sequence aligner COATi can align protein-coding regions in the presence of artifacts with higher accuracy than current methods, the execution time required for sequences longer than a few thousand nucleotides can be a limiting factor. To address this limitation, I developed an approximate model of the core evolutionary model (triplet) that can be implemented using standard dynamic programming techniques and speeds up the alignment operation to execution times comparable with popular aligners. In this chapter, I have undertaken a comprehensive exploration of the modal and marginal models, providing a detailed description of their definitions and assessing their accuracy in approximating the evolutionary processes of the triplet model.

The evaluation of results across various branch lengths highlights the remarkable fidelity of the marginal model to the triplet model, with similar outcomes in average alignment error and the number of perfect alignments. Conversely, the performance of the modal model, while comparable for short branch lengths, gradually diminishes as branch lengths increase. This decline in performance can be attributed to how the substitution probabilities are handled over evolutionary time, as per the definition of the model. Consequently, I recommend employing the triplet model for achieving the highest accuracy, especially when sequence lengths and computational resources permit. However, in cases where these factors pose a limitation, the marginal model, accompanied by its dynamic programming alignment approach, is a robust alternative.

The analysis described in this chapter can be further improved by adjusting gap opening rates in the sequence evolution simulator according to changes in branch length. The algorithm used in this chapter has a fixed gap opening parameter g that does not scale with branch length. In addition, the simulation algorithm should implement different rates for insertion and deletion events, aligning more closely with the prevalent patterns often observed in biological data (Zhang and Gerstein, 2003; de Jong and Rydén, 1981).

Future work includes developing an algorithm that can search alignment space to improve the initial multiple-sequence alignment that COATi can currently produce. This will allow COATi to improve the results on the accurate alignment inference of protein-coding regions in the presence of artifacts, a pressing issue in modern computational biology.

Chapter 4

ALNDOTPLOT: VISUAL REPRESENTATION AND ANALYSIS OF PAIRWISE ALIGNMENTS

4.1 Introduction

The analysis of biological sequences is the inference of unique and unobservable evolutionary events at the DNA level (Morrison, 2018). Alignment inference is an essential step required to address questions across multiple branches of biology, including molecular biology, microbiology, and ecology. The field has seen substantial progress since modern sequence alignment began with the computer-adaptable algorithm of Needleman and Wunsch (1970), replacing the arduous manual arrangement of residues as the default method for sequence alignment.

The development of sequence alignment methods has experienced a continuous effort to improve both their accuracy and speed. The quality of sequence alignments directly impacts the reliability of downstream analyses, and thus, alignment evaluation plays a pivotal role in quality control. Existing evaluation methods can be divided into metrics and scores that summarize similarities between alignments and tools that quantify the uncertainty associated with each column of an alignment. A limitation of these methods is that they can compare a summary metric between two alignments or provide site information about an alignment, but cannot combine both approaches. While aligners typically report a single best result using an evolutionary or scoring model, equally optimal (equivalent) alignments often exist and are rarely reported. However, equivalent alignments and suboptimal alternatives are common in sampling and multiple sequence refinement. Being able to perform a thorough

comparison between such alignments can be very valuable for understanding how the underlying biological models work. This can also lead to more detailed comparisons of different methods in validating sequence aligners, improve the algorithms that search for suboptimal alignments in multiple sequence refinement, and better assess sampling results.

4.1.1 Alignment Accuracy

Popular alignment similarity scores include sum-of-pairs (SP) and total column score (TC). SP is the percentage of correctly aligned residue pairs in an alignment, which measures how well pairs of sequences are aligned, while TC is the percentage of correctly aligned columns in an alignment, testing the ability to align all sequences correctly (Thompson *et al.*, 2005). In the case of pairwise alignment, both metrics are identical. A more informative set of metrics that consider indels and the evolutionary history of events in a phylogenetic tree was put forth by Blackburne and Whelan (2011). These metrics range from zero to one and can be interpreted as the probability that a randomly selected residue will be aligned to a different location against a sequence that does not contain such residue. Notably, similarity and distance metrics are easily scalable to compare results over large datasets. However, they struggle to find specific portions where alignments diverge and what are their differences.

Tools and methods that quantify alignment uncertainty include Heat or Tails (Landan and Graur, 2007), a method that considers the probability distribution of possible placements for each sequence within a multiple sequence alignment; GUIDANCE (Penn *et al.*, 2010), which uses bootstrap algorithms; GUIDANCE 2 (Sela *et al.*, 2015), which combines three sources of uncertainty: co-optimal solutions, guide tree instability, and opening gap penalty; ZORRO (Wu *et al.*, 2012), based on hidden Markov models; and MUMSA (Lassmann and Sonnhammer, 2005), which calculates

the portion of identically aligned regions; and posterior decoding, a popular method used in the context of Markov models that calculates the probability of each state in the alignment path and is particularly useful when many different paths have almost the same probability as the most likely one. These methods provide information about the reliability of each column in an alignment and identify uncertain sections, allowing researchers to remove them to prevent errors that may bias downstream analysis. While these metrics can help improve the quality of genomic pipeline results, they are also limited to analyzing one alignment at a time.

4.1.2 Alignment Visualization

A common feature shared among the alignment uncertainty applications described above is their display of confidence scores and the alignment in matrix form (Fig. 4.1). This format of alignment visualization provides a representation of an alignment, where residues are typically colored by type and are supported by numerous software packages and web-based tools (e.g., Zhou *et al.*, 2022; Yachdav *et al.*, 2016). A matrix representation of an alignment displays two or more aligned sequences as rows, with each column representing a site. However, while this format is intuitive, it does not allow comparing two or more alignments.

```
CAT AAG CGG TCG GAC ---
CAG CAG CGG TCG GAC ACG
CAT CAG CGG TC- --C ACA
CAT CAG CCG TCG GAC ACG
```

Figure 4.1: Example of a DNA multiple sequence alignment visualization in matrix form. Every row corresponds to a sequence and every column is a position in the alignment. Nucleotides are colored by type and gaps are shown in black.

Dot plots are an additional visual representation tool used to compare two sequences that facilitate the identification of similarities, differences, and underlying

patterns, introduced by Gibbs and McIntyre (1970). They consist of a matrix where one sequence is displayed on the x-axis, left to right, and the other on the y-axis, top to bottom, and using dots indicate matching residues between the sequences. Conversely, mismatching nucleotides are left blank. Note that gap symbols are typically not considered in dot plots as they evaluate unaligned sequences.

In biological sequence analysis, dot plots are suitable for identifying regions of similarity between sequences, evidenced by diagonal lines of dots that run left to right and top to bottom. The length and pattern of these contiguous dots provide insight into the nature and extent of the similarity. In addition, dot plots are used to identify specific biological events, including repeated regions, tandem repeats, palindromic regions, and microsatellite patterns (Fig. 4.2). This illustrates that dot plots are a simple and versatile tool for comparing pairs of unaligned sequences.

4.1.3 Comparison of Pairwise Alignments

The tools described above provide valuable and scalable algorithms for measuring distance or scoring similarity between alignments and methods for identifying uncertain regions within alignments. However, they cannot provide detailed information about how and where alignments differ. To fill this gap, I have developed AlnDotPlot, a visual tool inspired by traditional dot plots that can compare alternative pairwise alignments and is available as an R package. AlnDotPlot can provide a detailed comparison between a handful of pairwise alignments, identify patterns in hundreds of alignment sampling results, and find short base pair differences among alignments of a few kilobases. This software combines the simple nature of dot plots with features to analyze pairwise sequence alignments.

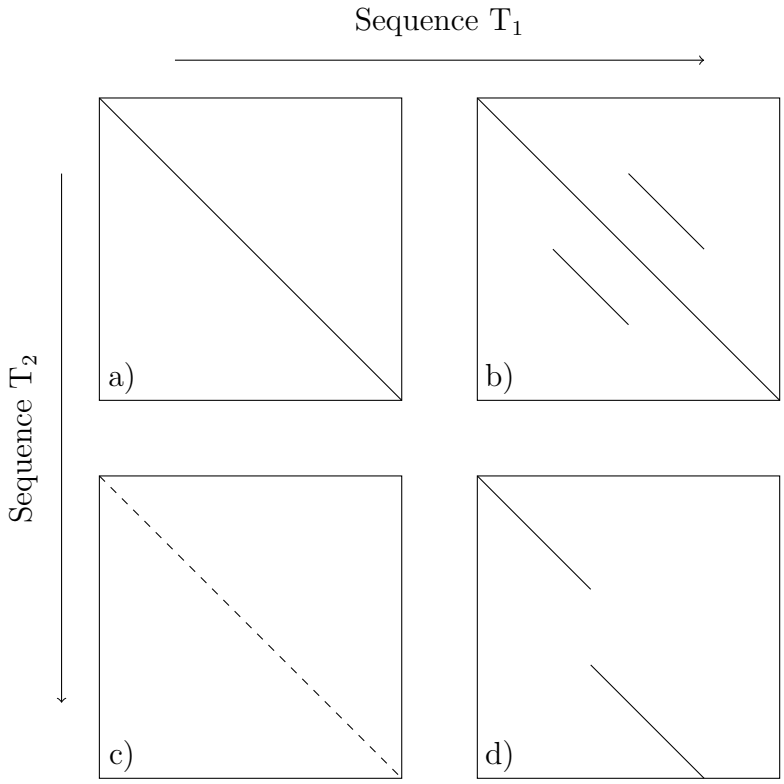


Figure 4.2: . Overview of characteristic patterns in dot plots. **a)** A continuous main diagonal shows perfect similarity. **b)** Parallels to the main diagonal indicate repeated regions on different parts of the sequences. **c)** When the diagonal is a discontinuous line this indicates that the sequences T_1 and T_2 share a common ancestor. **d)** Partial deletion in sequence T1 or insertion in sequence T2. Work modified from Schulz *et al.*, 2008

4.2 Implementation

AlnDotPlot can generate alignment dot plots that compare sets of pairwise alignments. This R package can read in alignments in FASTA format and produce results in PDF and TEX format. Internally, the alignment information is converted into matrices, which are in turn, used to create the dot plots. I use the *tikzDevice* R package to create the final results for its versatility and accuracy (Sharpsteen *et al.*, 2023). The following sections describe the different models and their implementation and use cases. AlnDotPlot is available as an R software package at

`https://www.github.com/jgarciamesa/alndotplot`.

4.2.1 From Alignment to Dot Matrix

AlnDotPlot generates different alignment dot plots given a set of pairwise alignments. To generate these figures, the first step is to read in the pairwise alignments. Next, the alignment information must be transformed into a dot matrix, which stores this information in a conventional two-dimensional matrix. Based on the different model designs, I have implemented two types of dot matrices. For the simplest model, a traditional dot plot, the dot matrix records only the positions within the alignments involving substitutions (matches and mismatches). Given two sequences s and v, this dot matrix has dimensions $|s|$ by $|v|$, where each row corresponds to symbols in s and every column to symbols in v. Every cell in the matrix represents a symbol from s that matches (or mismatches) with a symbol in v. The value in every cell is the count of substitutions between these symbols across all input alignments. Figure 4.3 showcases a sample alignment and its corresponding traditional dot matrix.

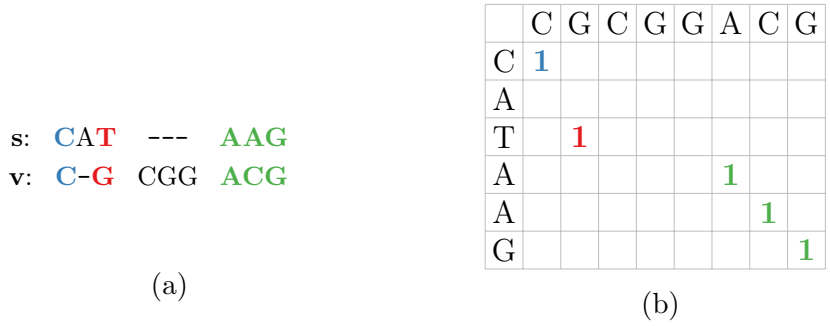


Figure 4.3: Converting a pairwise alignment to a traditional dot matrix. (a) is an alignment of sequences s and v, with matches and mismatches (substitutions) colored arbitrarily. (b) is a matrix with the characters in sequence s as rows and the characters in sequence v as columns. The values indicate the count of substitutions between the row and column nucleotides (i.e., the first 'C' in both sequences is matched once on the alignment). Note empty cells have a count of zero, omitted here.

Expanded dot matrices extend traditional dot matrices, described above, by incor-

porating indel information, and a similar construction procedure. Given a collection of pairwise alignments between sequences s and v, this dot matrix has dimensions $(2 \cdot |s| + 1)$ by $(2 \cdot |v| + 1)$, where even rows and columns correspond the symbols in s and v respectively, and odd rows and columns represent gap symbols. Expanding upon the previous dot matrix design, this configuration enables marking insertion and deletion events with clarity. The values within the matrix represent the count of the corresponding row and column symbol pairs found in the alignments. Figure 4.4 illustrates how an expanded dot matrix is constructed.

Aln1
s: **CAT** **---** **AAG**
v: **C-G** CGG **ACG**

Aln2
s: **CA**T **--A** **A-G**
v: **CG-** CG**G** **A**C**G**

Aln3
s: **CAT** **--A** A-**G**
v: **CGC** GG**A** -C**G**

(a)

	-	C	-	G	-	C	-	G	-	G	-	A	-	C	-	G	-
-	3																
C		3															
-			3														
A			1	2													
-			1		2												
T				1	1	1											
-					2	2	3	3	3	2	2						
A										1		2					
-											1		2				
A												1	1	1			
-													2	2	3		
G																3	
-																	3

(b)

Figure 4.4: Converting three alternative alignments to an expanded dot matrix. (a) is three alternative alignments of sequences s and v. (b) is a matrix with the characters in sequence s as rows and the characters in sequence v as columns. In addition, gap symbols have been added to represent indel information. The values indicate the count of substitutions or indels between the row and column characters (i.e., the first 'C' in both sequences is matched in all three alignments). Note empty cells have a count of zero, omitted here.

4.2.2 Dot Plot Models

Traditional Model

The traditional dot plot model resembles the original dot plots most, adapting the idea of identifying similarity between sequences to pairwise alignments. Consequently, this model only considers substitutions by using the traditional dot matrix. After calculating the counts for each cell, these are converted into frequencies. These values represent the percentage of alignments where each pair of symbols is aligned together. The final step is to draw squares on the tikz grid filled with a color corresponding to their frequency.

Figure 4.5 illustrates a traditional dot plot with three different alignments. Diagonal rows of squares, running left to right and top to bottom, indicate regions of contiguous substitutions. In addition, darker sections are most common among the alignments, while lighter squares indicate less frequent pairings. This model retains the simplicity of dot plots and is an efficient tool for highlighting common substitution patterns in pairwise alignments. Furthermore, the traditional model is scalable to a large number of sequences.

Aln 1

CAT AAG CGG TCG GAC ---
CAG CGG TCC CCG GAC ACG

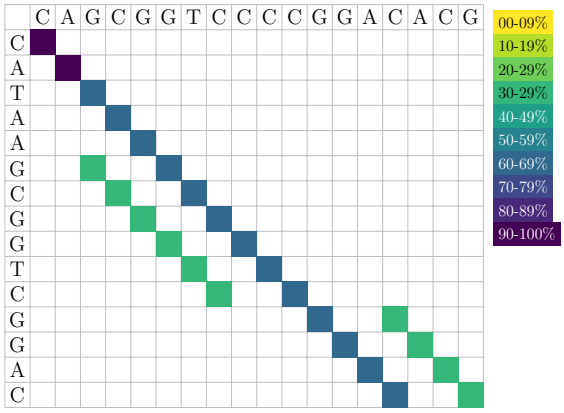
Aln 2

CAT AAG CGG TC- --G GAC ---
CA- --G CGG TCC CCG GAC ACG

Aln 3

CAT AAG CGG TC- --G GAC
CAG CGG TCC CCG GAC ACG

(a)



(b)

Figure 4.5: Example of a traditional alignment dot plot of three possible pairwise alignments. Squares represent matching and mismatching nucleotides, while the color gradient indicates their frequency in the alignments. The first two nucleotides 'C' and 'A' are matched in all three alignments (100%). The remaining squares on the main diagonal are only shared in two alignments (66%), while the remaining squares are only present in one alignment (33%).

Expanded Model

The expanded model includes information about indel events and therefore uses an expanded dot matrix to store alignment counts. These plots are conceptually similar to conventional visual aids to explain Needleman-Wunsch (Needleman and Wunsch, 1970) or Gotoh (Gotoh, 1982) algorithms where an alignment is illustrated as the path through a matrix. In comparison to the previous model, these plots add gap symbols between the characters of each sequence and are used to mark indel events. However, the process of creating the tikz grid with squares is similar. Counts are converted to frequencies and used to fill the squares in the matrix with the appropriate color. This model can easily accommodate large amounts of alignments.

Figure 4.6 illustrates how an expanded dot plot is created from the same three pairwise alignments as in the previous model. In addition to the diagonals of squares that indicate substitutions, this plot provides information about how they are con-

Aln 1

CAT	AAG	CGG	TCG	GAC	---
CAG	CGG	TCC	CCG	GAC	ACG

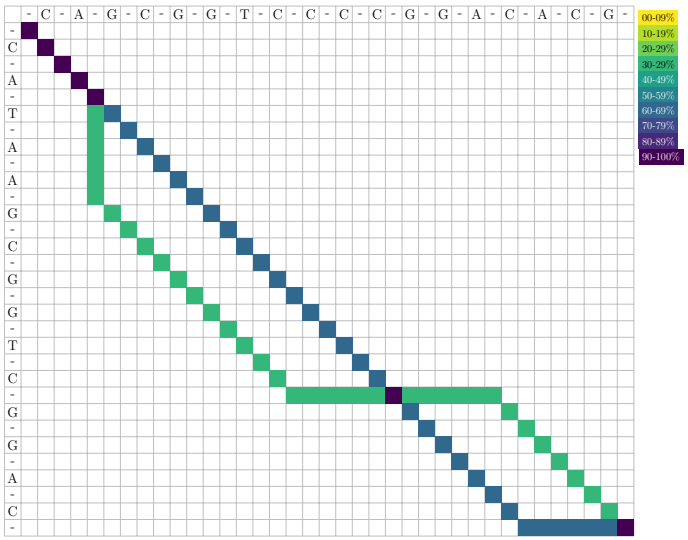
Aln 2

CAT	AAG	CGG	TC-	--G	GAC	---
CA-	--G	CGG	TCC	CCG	GAC	ACG

Aln 3

CAT	AAG	CGG	TC-	--G	GAC
CAG	CGG	TCC	CCG	GAC	ACG

(a)



(b)

Figure 4.6: Example of an expanded alignment dot plot for three possible pairwise alignments. Squares indicate substitution, insertion, or deletion events between symbols of the sequences. Diagonal rows of squares represent substitutions, while vertical and horizontal contiguous squares represent deletions and insertions, respectively. Colors indicate the frequency of events in the alignments.

nected. Furthermore, the expanded model uncovers a high-traffic section where paths cross, a cell connecting two gap symbols with 100% frequency.

Line Model

The line model introduces a new concept of nodes connected with edges to form a path or line. The nodes represent events in the alignment, while the edges connect pairs of cells in an alignment matrix. This model is an evolution of the expanded model, where the events are alignment events are marked and the edges clarify the transition between them. The line in a plot can split into different branches, indicating sections where the alignments are different. An expanded dot matrix is used to build a line dot plot. Similarly to the previous models, counts are converted to frequencies and used to fill the nodes with the corresponding color. However, the line model utilizes

three matrices to store substitutions, insertions, and deletions separately, needed to identify source and destination information to plot the edges. In addition, the edges are colored with the frequency of the destination node. This model can be seen as an extension of the expanded model that unveils information about high-traffic nodes and introduces the concept of a path, which will be useful later. Furthermore, this model is easily scalable to large amounts of data.

Figure 4.7 illustrates the line alignment dot plot resulting from three alignments. In comparison to the expanded model, figure 4.7-b helps disentangle the alignment path in congested areas such as the node where the three alignments, represented by two different branches, cross.

Aln 1

CAT AAG CGG TCG GAC ---
CAG CGG TCC CCG GAC ACG

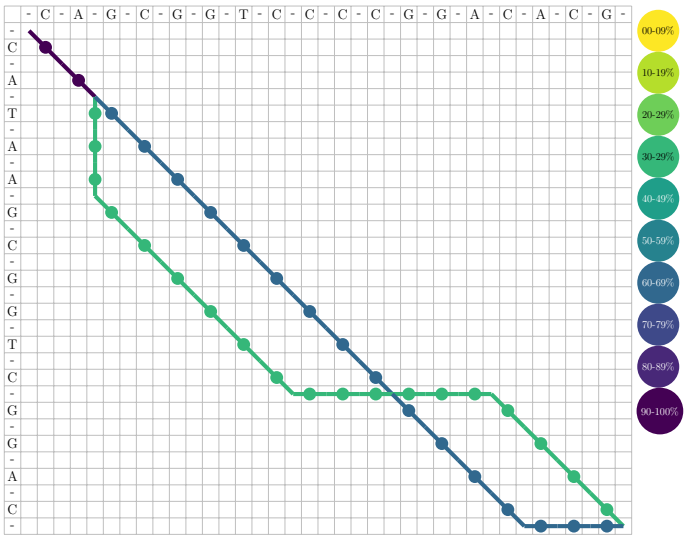
Aln 2

CAT AAG CGG TC- --G GAC ---
CA- --G CGG TCC CCG GAC ACG

Aln 3

CAT AAG CGG TC- --G GAC
CAG CGG TCC CCG GAC ACG

(a)



(b)

Figure 4.7: Example of a line alignment dot plot for three possible pairwise alignments. Nodes represent substitution, insertion, or deletion events between characters in the sequences or gap symbols. Edges join nodes to indicate a path or line, and the colors specify the frequency of events in the alignments.

Multiple Lines Model

The multiple lines model provides the most information about a set of pairwise alignments and extends the line model by encoding each alignment as a distinct path. Events for every input alignment are stored into their own three matrices, one for each of substitutions, deletions, and insertions, resulting in a collection of matrix trios. This model converts the list of matrix counts into frequencies and either colors each alignment with the frequency in the set of input alignments, or colors each alignment differently. A novel feature of this model is that every alignment has a designated position on each cell preventing different alignments from overlapping.

The multiple lines dot plot resulting from the three alignments used throughout this section is finally able to fully distinguish the different alignments (Fig. 4.8). This provides a detailed comparison of the alignments.

Aln 1

CAT AAG CGG TCG GAC ---
CAG CGG TCC CCG GAC ACG

Aln 2

CAT AAG CGG TC- --G GAC ---
CA- --G CGG TCC CCG GAC ACG

Aln 3

CAT AAG CGG TC- --G GAC
CAG CGG TCC CCG GAC ACG

(a)

(b)

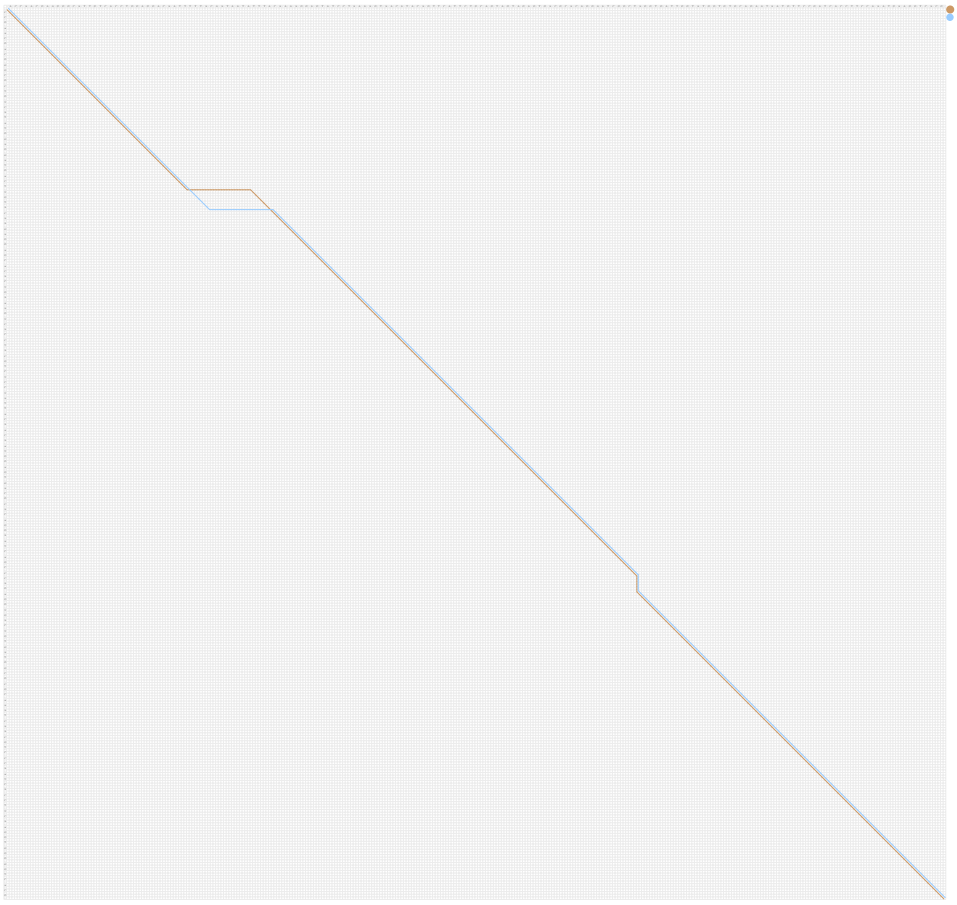
Figure 4.8: Example of a multiple lines dot plot for three possible pairwise alignments. Nodes represent substitution, insertion, or deletion events and edges connect them. In this model, all three alignments are different and therefore are displayed as unique paths. This allows full distinction of each alignment, providing a detailed comparison.

4.2.3 Bubble Finding

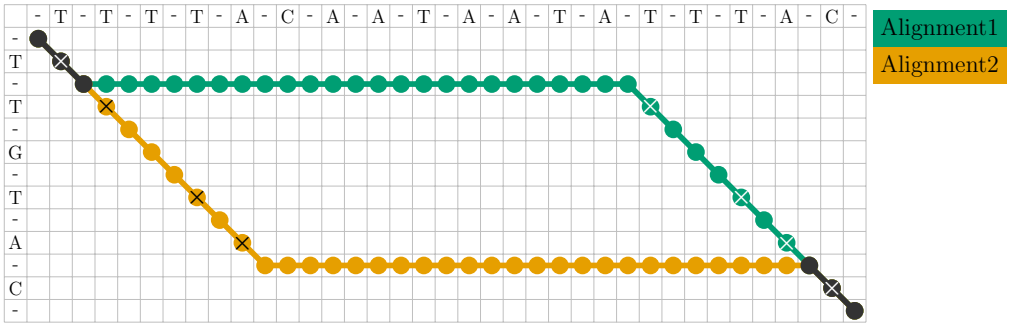
The models described in the previous section are designed to handle large amounts of sequences. However, given the nature of the implementation, properly plotting long

sequences can be challenging; as illustrated by a multiple lines dot plot of two pairwise alignments with 180 nucleotides long sequences (Fig. 4.9-a). To improve the abilities of AlnDotPlot, I have developed a feature that, given two pairwise alignments, can detect sections of dissimilarity and plot them using the line model (Fig. 4.9-b). In addition, the resulting dot plots explicitly mark matches with an 'X' on the corresponding nodes. The bubble finding function finds all sections where the alignments diverge and creates a line dot plot for each one. This feature is specially designed for alignments of long sequences with relatively small differences, allowing a more precise evaluation.

The algorithm to find the divergent sections, also called bubbles, conceptually classifies the relationship between nodes as parent and child. A parent node precedes the current one, while the child node follows it, ordered from the top-left to the bottom-right of the matrix. Note that nodes can have multiple children and parents in the line model. The algorithm first creates a list of full nodes, which are those shared by all alignments. Then, using this list, the function looks for the first node with a full parent and without a full child, indicating the beginning of a bubble. Conversely, the end of the bubble is marked by the next node with no full parents and a full child node. This is repeated until the end of the line, carefully avoiding misclassifying the start and end nodes of the alignment. In addition, the algorithm adjusts the start and end nodes in the bubble to display entire codons. Figure 4.10 showcases line dot plots found using the bubble-finding function of two pairwise alignments with 2.3 kilobases long sequences from the results of chapter 3.



(a)



(b)

Figure 4.9: Example of a multiple lines dot plot of two pairwise alignments with two sequences of approximately 180 nucleotides each (a) and the result of running the bubble finding function over the same alignments (b). Rendering dot plots of long sequences is limiting. However, the AlnDotPlot can find the difference in the alignment and create a line plot only of that section.

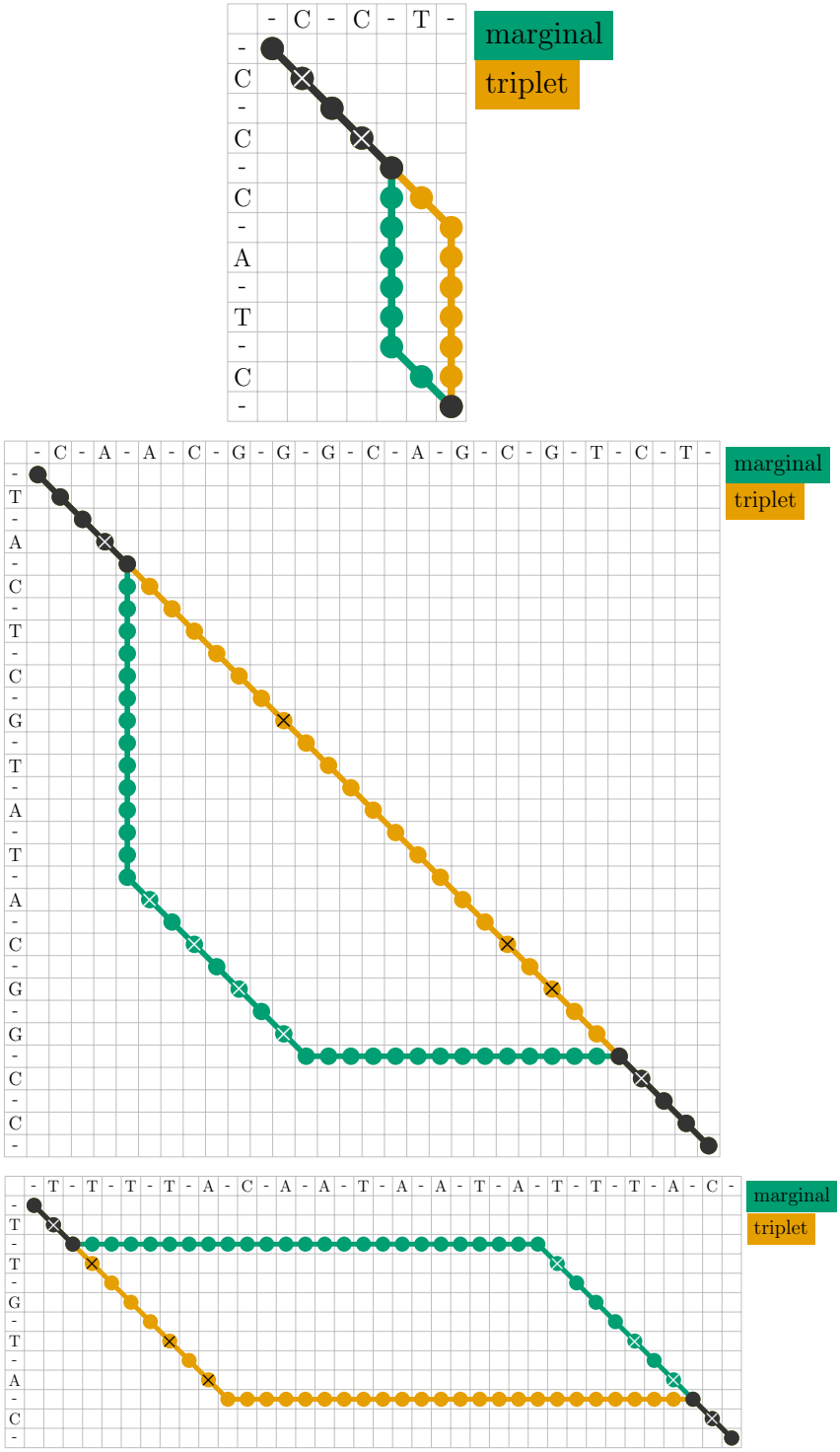


Figure 4.10: Line dot plots obtained using the bubble finding from two different pairwise alignments of approximately 2.3 kilobases long sequences. This showcases the bubble-finding algorithm. Green paths correspond to the COATi marginal-mg-sum model and orange paths to the COATi triplet-mg model. Match nodes are crossed in black or white.

4.2.4 Codon Similarity

In addition to the nucleotide interactions in the alignment, I am interested in understanding how codon similarity can affect the alignment path. Therefore, AlnDotPlot can color code grid sections in the dot plot to represent the affinity of the two intersecting codons (row and column codons). This is done by dividing the grid into 6 by 6 submatrices (3 nucleotides per codon plus gap symbols) and coloring their background. There are four different colors, indicating different similarity levels. First, if the two codons are identical, meaning no changes at the DNA level, the section is colored green. Second, if the codons code for the same amino acid, also known as synonymous, their cross-section is colored yellow. The third and fourth categories classify the non-synonymous codons interactions based on their evolutionary properties as defined by Kosiol et al. (2007). Using these categories if the amino acid translation for both codons belongs to the same group, the section is filled in orange; otherwise, the intersecting region is colored in red. This contextualizes the alignment and can provide insight into why certain models avoid specific areas (e.g., Fig. 4.11).

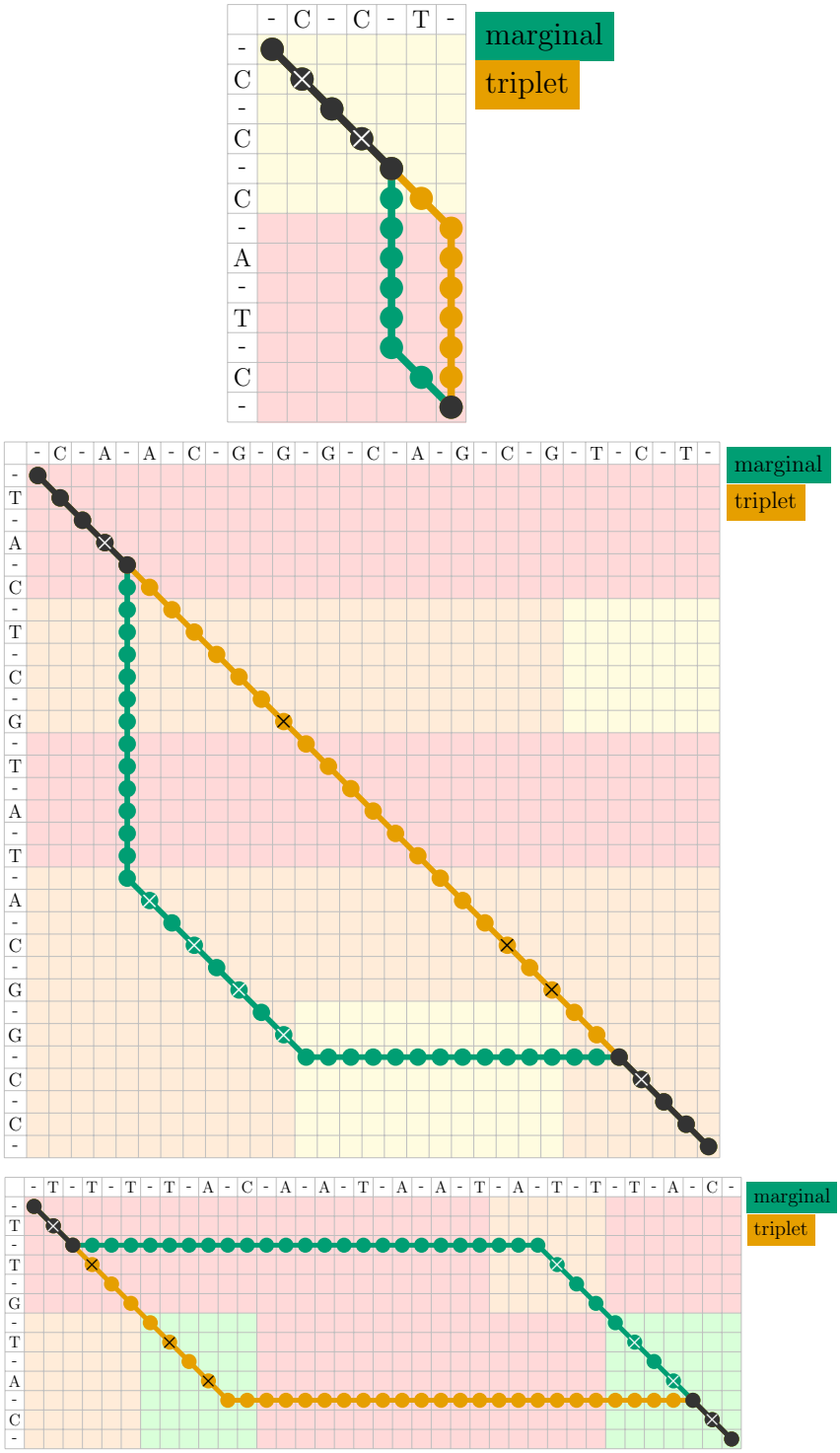


Figure 4.11: Line alignment dot plots with codon similarity information. Equal codon interactions are colored in green and different codons that code for the same amino acid are in yellow. Codons encoding for amino acids that are in the same similarity group are in orange, otherwise in red.

4.3 Discussion

The development of this tool responds to a need to understand better the difference between two alignment models with similar benchmark results. While the simpler models might not fit many scenarios, the multiple lines model and the bubble-finding feature are valuable tools in analyzing sequence alignments. AlnDotPlot can compare pairwise alignments and provide detailed insights through an intuitive visual representation. This R package can be included in alignment assessment pipelines in addition to distance metrics and alignment accuracy tools.

Future work includes supporting other output formats, such as image files that support rasterization to convert from plots back to alignment. In addition, developing a method to display long pairwise alignments would address a clear limitation.

Chapter 5

CONCLUSION

Sequence alignment is an essential method in bioinformatics, serving as the foundation for many genomic analyses. Overlooking artifacts and errors during alignment reconstruction can impact downstream analyses, potentially leading to inaccurate findings in comparative and functional genomic studies. While such errors are eventually fixed in the reference genomes of model organisms through a laborious process, large amounts of genomic data used by researchers still contain these artifacts, often prompting the discarding of valuable data to prevent them from impacting results.

In this work, I designed, developed, and evaluated a novel statistical, codon-aware pairwise sequence aligner tailored to address common artifacts found in protein-coding genes. In chapter 2, I explained the aligner and its underlying evolutionary model, along with a comparative evaluation against conventional pairwise aligners, using human and gorilla homologous protein-coding sequences. Remarkably, despite humans and gorillas being two relatively close species, the results of my aligner demonstrated significant improvement. However, it is worth noting that the model exhibited a limiting computational runtime cost with sequences exceeding a few kilobases. Therefore, in chapter 3, I presented and evaluated an approximate model that maintained similar accuracy while achieving competitive execution times. In the subsequent chapter 4, I described a software package used for analyzing the alignment results from the previous chapter.

In the present era, the volume of genomic data generated is staggering, yet often remains unpolished due to the labor-intensive nature of genome curation. As a result, bioinformatic tools must continually evolve to incorporate models capable of

addressing common artifacts. The ongoing development of software and statistical methodologies for biological sequence analysis will remain crucial, as they serve as the bridge transforming raw data into meaningful insights.

Bibliography

Abascal, F., R. Zardoya and M. J. Telford, "Translatorx: multiple alignment of nucleotide sequences guided by amino acid translations", Nucleic acids research **38**, W7–W13 (2010).

Allauzen, C., M. Riley, J. Schalkwyk, W. Skut and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library", in "International Conference on Implementation and Application of Automata", pp. 11–23 (Springer, 2007).

Aniba, M. R., O. Poch and J. D. Thompson, "Issues in bioinformatics benchmarking: the case study of multiple sequence alignment", Nucleic Acids Research **38**, 21, 7353–7363, URL `https://doi.org/10.1093/nar/gkq625` (2010).

Bininda-Emonds, Olaf, "transalign: using amino acids to facilitate the multiple alignment of protein-coding dna sequences", BMC bioinformatics **6**, 1, 1–6 (2005).

Blackburne, B. P. and S. Whelan, "Measuring the distance between multiple sequence alignments", Bioinformatics **28**, 4, 495–502, URL `https://doi.org/10.1093/bioinformatics/btr701` (2011).

Bradley, R. K. and I. Holmes, "Transducers: an emerging probabilistic framework for modeling indels on trees.", Bioinformatics **23**, 23 (2007).

Cartwright, R. A., "Dna assembly with gaps (dawg): simulating sequence evolution", Bioinformatics **21**, Suppl_3, iii31–iii38 (2005).

Cartwright, R. A., "Logarithmic gap costs decrease alignment accuracy", BMC bioinformatics **7**, 1, 1–12 (2006).

Charif, D. and J. Lobry, "SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis.", in "Structural approaches to sequence evolution: Molecules, networks, populations", edited by U. Bastolla, M. Porto, H. Roman and M. Vendruscolo, Biological and Medical Physics, Biomedical Engineering, pp. 207–232 (Springer Verlag, New York, 2007), ISBN : 978-3-540-35305-8.

Cornish-Bowden, A., "Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984.", Nucleic acids research **13**, 9, 3021 (1985).

de Jong, W. W. and L. Rydén, "Causes of more frequent deletions than insertions in mutations and protein evolution", Nature **290**, 5802, 157–159 (1981).

Delport, W., K. Scheffler and C. Seoighe, "Models of coding sequence evolution", Briefings in bioinformatics **10**, 1, 97–109 (2009).

Durbin, R., S. R. Eddy, A. Krogh and G. Mitchison, *Biological sequence analysis: probabilistic models of proteins and nucleic acids* (Cambridge university press, 1998).

Fletcher, W. and Z. Yang, "Indelible: a flexible simulator of biological sequence evolution", Molecular biology and evolution **26**, 8, 1879–1888 (2009).

Fletcher, W. and Z. Yang, "The effect of insertions, deletions, and alignment errors on the branch-site test of positive selection", Molecular biology and evolution **27**, 10, 2257–2267 (2010).

Gibbs, A. J. and G. A. McIntyre, "The diagram, a method for comparing sequences: Its use with amino acid and nucleotide sequences", European journal of biochemistry **16**, 1, 1–11 (1970).

Gotoh, O., "An improved algorithm for matching biological sequences", Journal of molecular biology **162**, 3, 705–708 (1982).

Holmes, I. and W. J. Bruno, "Evolutionary hmms: a bayesian approach to multiple alignment", Bioinformatics **17**, 9, 803–820 (2001).

Holmes, I. and G. Rubin, "An expectation maximization algorithm for training hidden substitution models", Journal of molecular biology **317**, 5, 753–764 (2002).

Hubbard, T., D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down *et al.*, "The ensembl genome database project", Nucleic acids research **30**, 1, 38–41 (2002).

Hubisz, M. J., M. F. Lin, M. Kellis and A. Siepel, "Error and error mitigation in low-coverage genome assemblies", PloS one **6**, 2, e17034 (2011).

Jackman, S. D., L. Coombe, J. Chu, R. L. Warren, B. P. Vandervalk, S. Yeo, Z. Xue, H. Mohamadi, J. Bohlmann, S. J. Jones *et al.*, "Tigmint: correcting assembly errors using linked reads from large molecules", BMC bioinformatics **19**, 1, 1–10 (2018).

Katoh, K. and D. M. Standley, "Mafft multiple sequence alignment software version 7: improvements in performance and usability", Molecular biology and evolution **30**, 4, 772–780 (2013).

Kosiol, C., I. Holmes and N. Goldman, "An empirical codon model for protein sequence evolution", Molecular biology and evolution **24**, 7, 1464–1479 (2007).

Kullback, S. and R. A. Leibler, "On information and sufficiency", The annals of mathematical statistics **22**, 1, 79–86 (1951).

Landan, G. and D. Graur, "Heads or tails: a simple reliability check for multiple sequence alignments", Molecular biology and evolution **24**, 6, 1380–1383 (2007).

Lassmann, T. and E. L. Sonnhammer, "Automatic assessment of alignment quality", Nucleic acids research **33**, 22, 7120–7128 (2005).

Li, W.-H., "Unbiased estimation of the rates of synonymous and nonsynonymous substitution", Journal of molecular evolution **36**, 1, 96–99 (1993).

Löytynoja, A., "Phylogeny-aware alignment with prank", in "Multiple sequence alignment methods", pp. 155–170 (Springer, 2014).

Ly-Trong, N., S. Naser-Khdour, R. Lanfear and B. Q. Minh, "Alisim: a fast and versatile phylogenetic sequence simulator for the genomic era", Molecular biology and evolution **39**, 5, msac092 (2022).

Mächler, M., "Accurately computing log (1- exp (-— a—)) assessed by the rmpfr package", Tech. rep., Technical report (2012).

Morrison, D. A., "Is sequence alignment an art or a science?", Systematic Botany **40**, 1, 14–26 (2015).

Morrison, D. A., "Multiple sequence alignment is not a solved problem", (2018).

Muse, S. V. and B. S. Gaut, "A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome.", Molecular biology and evolution **11**, 5, 715–724 (1994).

Needleman, S. B. and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", Journal of Molecular Biology **48**, 3, 443–453 (1970).

Orlova, T. A., *Mathematical models, algorithms, and statistics of sequence alignment*, Ph.D. thesis, University of South Carolina (2010).

Penn, O., E. Privman, H. Ashkenazy, G. Landan, D. Graur and T. Pupko, "Guidance: a web server for assessing alignment confidence scores", Nucleic acids research **38**, suppl_2, W23–W28 (2010).

Raghava, G., S. M. Searle, P. C. Audley, J. D. Barber and G. J. Barton, "Oxbench: a benchmark for evaluation of protein multiple sequence alignment accuracy", BMC bioinformatics **4**, 1, 1–23 (2003).

Ranwez, V., S. Harispe, F. Delsuc and E. J. Douzery, "Macse: Multiple alignment of coding sequences accounting for frameshifts and stop codons", PloS one **6**, 9, e22594 (2011).

Rosenberg, M. S., *Sequence alignment: methods, models, concepts, and strategies* (Univ of California Press, 2009).

Schneider, A., A. Souvorov, N. Sabath, G. Landan, G. H. Gonnet and D. Graur, "Estimates of positive darwinian selection are inflated by errors in sequencing, annotation, and alignment", Genome biology and evolution **1**, 114–118 (2009).

Schulz, J., F. Leese and C. Held, "Introduction to dot-plots", URL http://www.code10.info/images/c10_files/algorithms/dot-plots/Fig%206%20-%20Schematische%20Dotplot%20Signaturen.gif (2008).

Sela, I., H. Ashkenazy, K. Katoh and T. Pupko, "Guidance2: accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters", Nucleic acids research **43**, W1, W7–W14 (2015).

Sharpsteen, C., C. Bracken, K. Müller, Y. Xie, R. Stubner and N. Bellack, "Package 'tikzdevice"', (2023).

Sievers, F., A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding *et al.*, "Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega", Molecular systems biology **7**, 1, 539 (2011).

Silvestre-Ryan, J., Y. Wang, M. Sharma, S. Lin, Y. Shen, S. Dider and I. Holmes, "Machine boss: rapid prototyping of bioinformatic automata", Bioinformatics **37**, 1, 29–35 (2021).

Statisticat and LLC., *LaplacesDemon: Complete Environment for Bayesian Inference*, URL https://web.archive.org/web/20150206004624/http://www.bayesian-inference.com/software, r package version 16.1.6 (2021).

Sullivan, J. and P. Joyce, "Model selection in phylogenetics", Annu. Rev. Ecol. Evol. Syst. **36**, 445–466 (2005).

Tavaré, S. *et al.*, "Some probabilistic and statistical problems in the analysis of dna sequences", Lectures on mathematics in the life sciences **17**, 2, 57–86 (1986).

Taylor, M. S., C. P. Ponting and R. R. Copley, "Occurrence and consequences of coding sequence insertions and deletions in mammalian genomes", Genome research **14**, 4, 555–566 (2004).

Thompson, J. D., P. Koehl, R. Ripp and O. Poch, "Balibase 3.0: latest developments of the multiple sequence alignment benchmark", Proteins: Structure, Function, and Bioinformatics **61**, 1, 127–136 (2005).

Whelan, S., P. I. de Bakker, E. Quevillon, N. Rodriguez and N. Goldman, "Pandit: an evolution-centric database of protein and associated nucleotide domains with inferred trees", Nucleic Acids Research **34**, suppl_1, D327–D331 (2006).

Wu, M., S. Chatterji and J. A. Eisen, "Accounting for alignment uncertainty in phylogenomics", PloS one **7**, 1, e30288 (2012).

Yachdav, G., S. Wilzbach, B. Rauscher, R. Sheridan, I. Sillitoe, J. Procter, S. E. Lewis, B. Rost and T. Goldberg, "Msaviewer: interactive javascript visualization of multiple sequence alignments", Bioinformatics **32**, 22, 3501–3503 (2016).

Yang, Z., "Estimating the pattern of nucleotide substitution", Journal of molecular evolution **39**, 105–111 (1994).

Yoon, B.-J., "Hidden markov models and their applications in biological sequence analysis", Current genomics **10**, 6, 402–415 (2009).

Zhang, C., Y. Zhao, E. L. Braun and S. Mirarab, "Taper: Pinpointing errors in multiple sequence alignments despite varying rates of evolution", Methods in Ecology and Evolution **12**, 11, 2145–2158 (2021).

Zhang, Z. and M. Gerstein, "Patterns of nucleotide substitution, insertion and deletion in the human genome inferred from pseudogenes", Nucleic acids research **31**, 18, 5338–5348 (2003).

Zhou, L., T. Feng, S. Xu, F. Gao, T. T. Lam, Q. Wang, T. Wu, H. Huang, L. Zhan, L. Li *et al.*, "ggmsa: A visual exploration tool for multiple sequence alignment and associated data", Briefings in Bioinformatics **23**, 4, bbac222 (2022).

Zhu, Z., *Profiling of Indel Phases in Coding Regions*, Ph.D. thesis, Arizona State University (2022).

APPENDIX A

SUPPLEMENTARY INFORMATION FOR CHAPTER 2

	dseq	Perfect	Best	Imperfect	F1 pos selection	F1 neg selection
tri-MG94	0.00221	5793	5139	1048	0.98073	0.99809
MAFFT	0.01471	5292	4692	1549	0.84314	0.98411
PRANK*	0.01828	4725	4774	2116	0.86749	0.98698
MACSE	0.01399	2861	3737	3980	0.79456	0.98199
ClustalΩ	0.02929	2893	2615	3948	0.68691	0.96938

* PRANK produced 42 empty alignments, calculations are based on 7719 alignments.

Table A.1: Accuracy of COATi codon-triplet-mg, PRANK, MAFFT, ClustalΩ, and MACSE on 7761 simulated sequence pairs. Perfect alignments have the same score as the true alignment, best alignments have lowest d_{seq}, and imperfect alignments have a different score than the true alignment when at least one method found a perfect alignment.

	dseq	Perfect	Best	Imperfect	F1 pos selection	F1 neg selection
tri-ECM	0.00238	5689	5045	1118	0.97803	0.99779
MAFFT	0.01451	5338	4677	1469	0.86048	0.98549
PRANK*	0.01903	4803	4851	2004	0.89250	0.98912
MACSE	0.01352	2903	3787	3904	0.82181	0.98359
ClustalΩ	0.02801	2979	2624	3828	0.72337	0.97244

* PRANK produced 69 empty alignments, calculations are based on 7692 alignments.

Table A.2: Accuracy of COATi codon-triplet-ecm, PRANK, MAFFT, ClustalΩ, and MACSE on 7761 simulated sequence pairs. Perfect alignments have the same score as the true alignment, best alignments have lowest d_{seq}, and imperfect alignments have a different score than the true alignment when at least one method found a perfect alignment.

	dseq	Perfect	Best	Imperfect	F1 pos selection	F1 neg selection
mar-MG94	0.00222	5808	5220	1075	0.97671	0.99766
MAFFT	0.01505	5301	4782	1582	0.85147	0.98455
PRANK*	0.01974	4856	5015	2027	0.89928	0.99000
MACSE	0.01429	2855	3893	4028	0.81569	0.98349
ClustalΩ	0.02870	2901	2610	3982	0.72399	0.97171

* PRANK produced 60 empty alignments, calculations are based on 7695 alignments.

Table A.3: Accuracy of COATi codon-marginal-mg, PRANK, MAFFT, ClustalΩ, and MACSE on 7755 simulated sequence pairs. Perfect alignments have the same score as the true alignment, best alignments have lowest d_{seq}, and imperfect alignments have a different score than the true alignment when at least one method found a perfect alignment.

	dseq	Perfect	Best	Imperfect	F1 pos selection	F1 neg selection
mar-ECM	0.00229	5781	5135	1081	0.97052	0.99710
MAFFT	0.01473	5379	4813	1483	0.85011	0.98491
PRANK*	0.01953	4830	4918	2032	0.87752	0.98790
MACSE	0.01400	2953	3893	3909	0.78977	0.98159
ClustalΩ	0.02918	2892	2611	3970	0.67847	0.96785

* PRANK produced 49 empty alignments, calculations are based on 7718 alignments.

Table A.4: Accuracy of COATi codon-marginal-ecm, PRANK, MAFFT, ClustalΩ, and MACSE on 7767 simulated sequence pairs. Perfect alignments have the same score as the true alignment, best alignments have lowest d_{seq}, and imperfect alignments have a different score than the true alignment when at least one method found a perfect alignment.

	dseq	Perfect	Best	Imperfect	F1 pos selection	F1 neg selection
Trip-MG	0.00113	3501	2890	309	0.99278	0.99932
MAFFT	0.00586	3162	2704	648	0.91064	0.99137
PRANK	0.00358	2829	2673	981	0.90332	0.99084
MACSE	0.00448	2552	2434	1258	0.87234	0.98857
ClustalΩ	0.02099	1772	1554	2038	0.75960	0.97686
Trip-MG-gor	0.00118	3463	2816	347	0.98993	0.99904

Table A.5: Accuracy of COATi codon-triplet-mg, PRANK, MAFFT, ClustalΩ, MACSE, and codon-triplet-mg with gorilla as the reference on 4003 of the 7761 simulated sequence pairs where the gorilla sequence was simulated without early stop codons. Perfect alignments have the same score as the true alignment, best alignments have lowest d_{seq}, and imperfect alignments have a different score than the true alignment when at least one method found a perfect alignment.

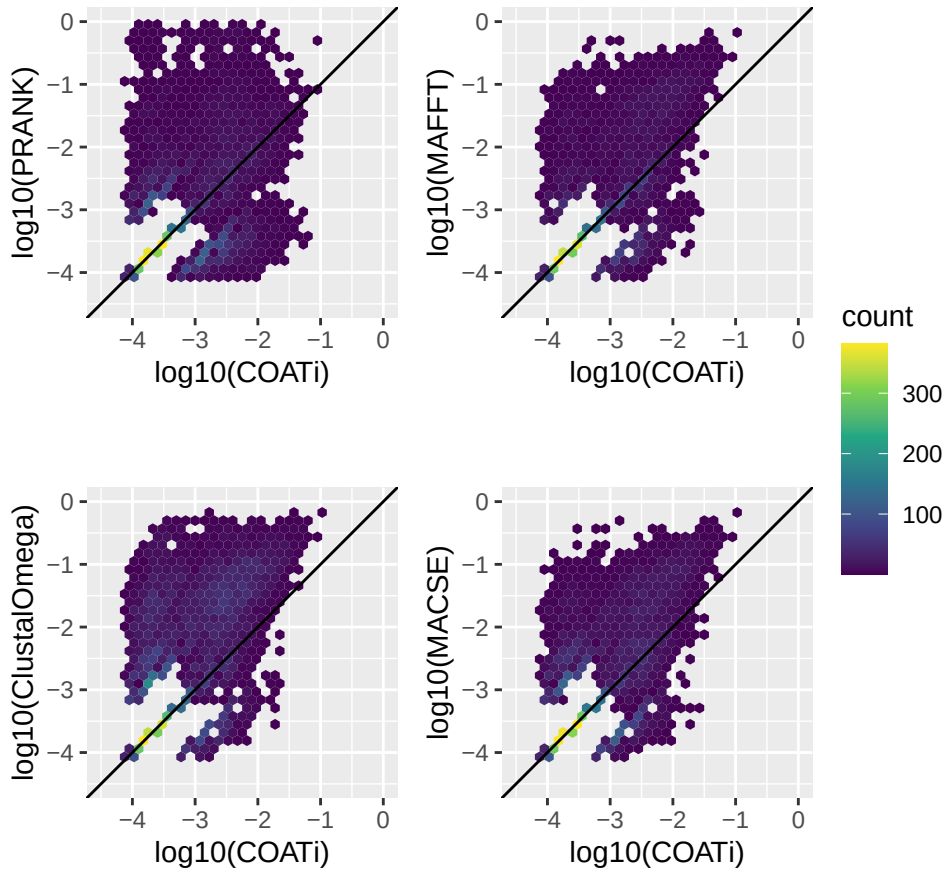


Figure A.1: Comparison of log10-transformed d_{seq} data with pseudocounts between COATi codon-triplet-mg and PRANK, MAFFT, ClustalΩ, and MACSE. COATi was significantly more accurate than other aligners; all p-values were $\leq 1.25e-76$.

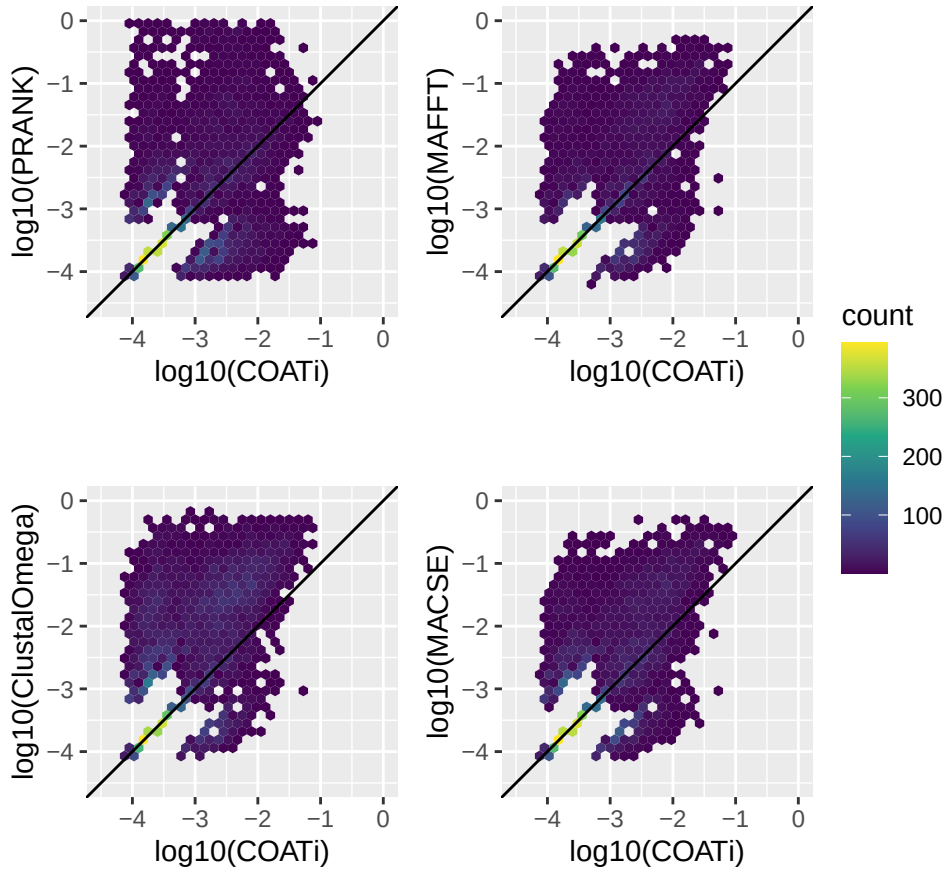


Figure A.2: Comparison of log10-transformed d_{seq} data with pseudocounts between COATi codon-triplet-ecm and PRANK, MAFFT, ClustalΩ, and MACSE. COATi was significantly more accurate than other aligners; all p-values were $\leq 3.23e-48$.

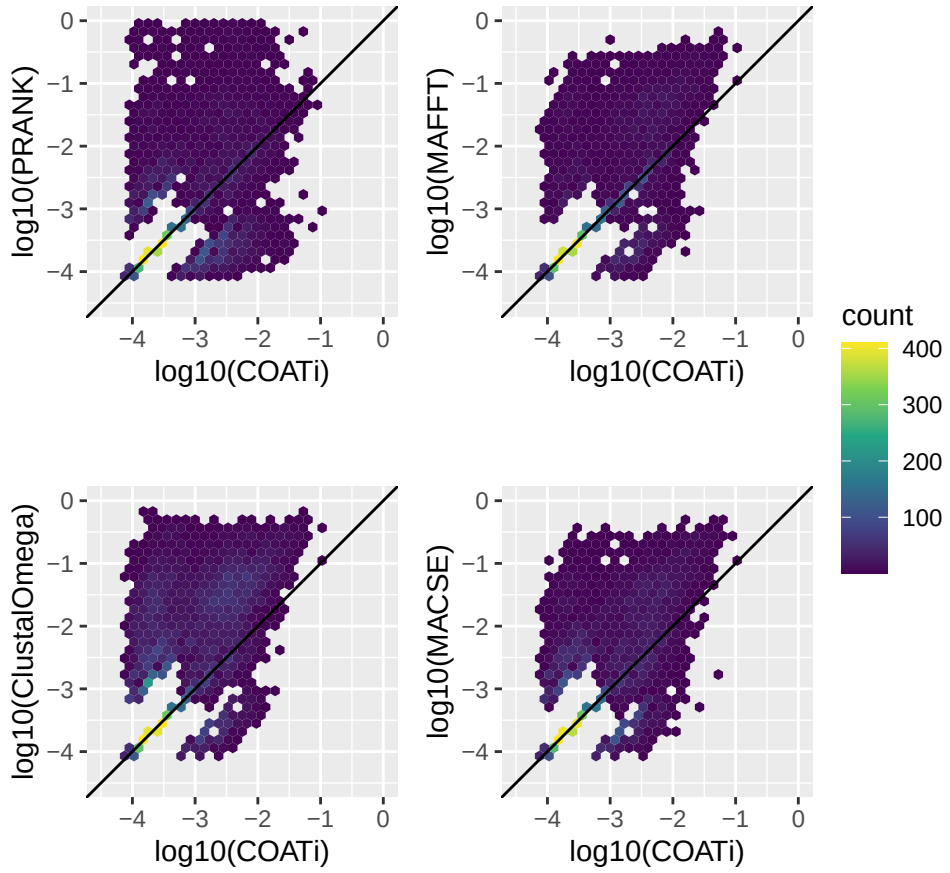
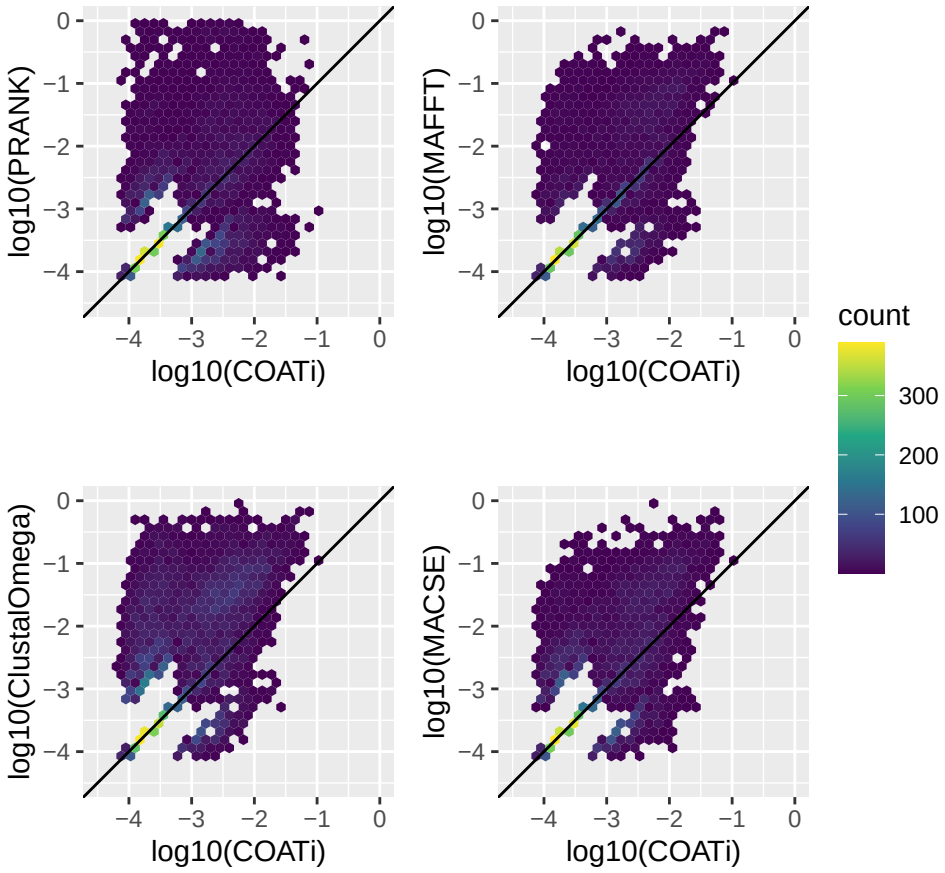


Figure A.3: Comparison of log10-transformed d_{seq} data with pseudocounts between COATi codon-marginal-mg and PRANK, MAFFT, ClustalΩ, and MACSE. COATi was significantly more accurate than other aligners; all p-values were $\leq 1.99e-53$.



![ht]

Figure A.4: Comparison of log10-transformed d_{seq} data with pseudocounts between COATi codon-marginal-ecm and PRANK, MAFFT, ClustalΩ, and MACSE. COATi was significantly more accurate than other aligners; all p-values were $\leq 1.44e-52$.

APPENDIX B

SUPPLEMENTARY INFORMATION FOR CHAPTER 3

Alignment Accuracy of the Triplet and Marginal ECM Models

t	Model	d_{seq}	Perfect alns	Imperfect alns
0.2	tri-ecm	4.075938e-04	11609	538
0.2	mar-ecm-sum	2.254086e-03	9568	2578
0.2	mar-ecm-max	1.153611e-02	3955	7941
0.4	tri-ecm	5.674235e-04	11146	439
0.4	mar-ecm-sum	2.782803e-03	9271	2333
0.4	mar-ecm-max	0.0330844	1751	9599
0.6	tri-ecm	7.325429e-04	10681	377
0.6	mar-ecm-sum	3.134104e-03	9007	2020
0.6	mar-ecm-max	7.975872e-02	669	10130
0.8	tri-ecm	9.450685e-04	10354	388
0.8	mar-ecm-sum	3.621644e-03	8862	1854
0.8	mar-ecm-max	1.771613e-01	197	10322
1.0	tri-ecm	1.202495e-03	9999	361
1.0	mar-ecm-sum	4.194538e-03	8646	1720
1.0	mar-ecm-max	3.459159e-01	55	10155

Table B.1: Table with complete results comparing results of triplet-ecm, marginal-ecm-max, and marginal-ecm-sum COATi models in aligning 13,758 simulated sequence pairs. The triplet-ecm model generates better alignments across all branch lengths. Best alignments have the lowest d_{seq}, perfect alignments have the same score as the true alignment or a zero d_{seq}, and imperfect alignments have a different score than true alignments when at least one model found a perfect alignment.

Benchmark Across Different CPUs

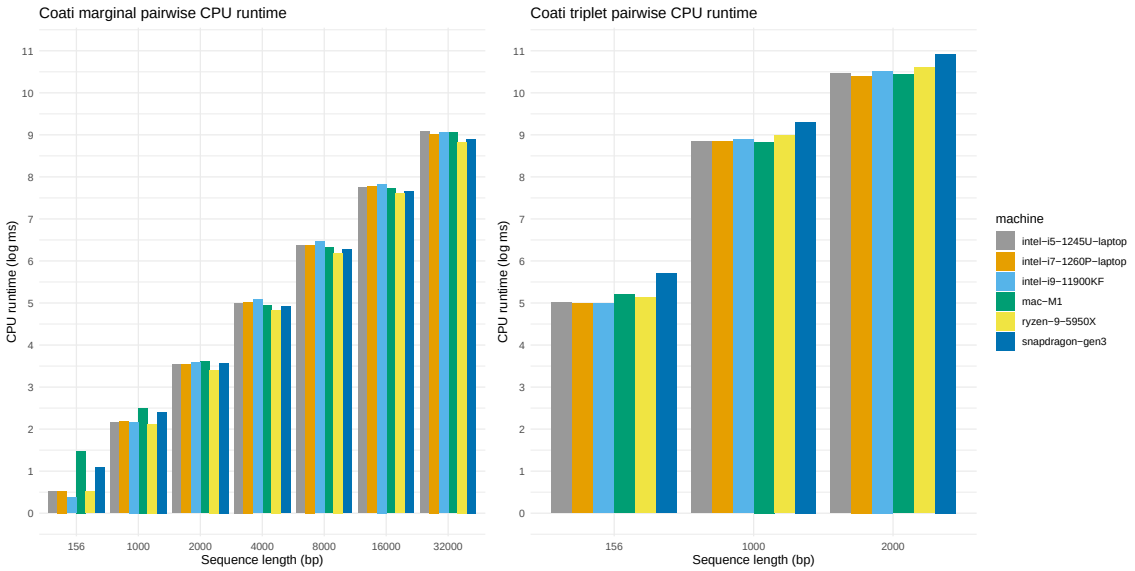


Figure B.1: Runtime benchmark of COATi marginal and triplet on different CPUs. Runtime is measured in seconds, while the benchmark consists of different length alignments.